ISSN 0542 9951

7
1996

NAJSTARSZE PISMO INFORMATYCZNE W POLSCE – UKAZUJE SIĘ OD 1965 ROKU

informatyka

miesięcznik profesjonalistów

P. 1877/96

Rada w radę o pracownikach

Lesław Wawrzonek

Kadry informatyczne w administracji państwowej i samorządowej

Spośród wielu kłopotów, jakie ma administracja rządowa z informatyką, do najpoważniejszych należy brak odpowiedniej kadry. Dlatego też w dokumentach rządowych określających przyszłość informatyzacji administracji państwowej szczególnie dużo miejsca poświęca się sprawom kadry mającej tworzyć tę przyszłość.

Janusz Maszkiewicz, przewodniczący Rady Koordynacyjnej ds. Informatyki przy RM i dyrektor Departamentu Informatyki URM twierdzi, że „z powodu braku specjalistów podjęto w przeszłości wiele nietrafnych i kosztownych decyzji". Nieprzygotowani merytorycznie pracownicy nie potrafią opracować zestawu wymagań dla systemu informatycznego, kontrolować przebiegu jego wdrożenia i usuwać usterek pojawiających się w czasie eksploatacji. Nie ma też zbyt wielu ludzi, którzy mogliby negocjować warunki cenowe z dostawcami krajowymi i zagranicznymi czy ustalać optymalne warunki techniczne zamawianych systemów informatycznych. Hohsztaplerom z „dobrą pieczątką" udało się na przykład sprzedać administracji państwowej za ponad 10 tys. zł projekt niedużego systemu informatycznego, który mógłby wykonać każdy student politechniki w ciągu tygodnia.

Oczywiście chodzi o pieniądze

W całej administracji państwowej z informatyką jest związanych 1200 osób. Praca w administracji publicznej stawia wysokie wymagania zarówno co do kwalifikacji, jak i przestrzegania zasad etycznych. Takich pracowników dotyczy też zakaz podejmowania dodatkowych zajęć, na które zresztą i tak nie mają czasu. W przeciwieństwie do naszej, rządowa administracja amerykańska ceni swoich informatyków i jest średnio trzecim w kraju pracodawcą (zależy to od stanu) pod względem wysokości płac, oferując od 5 do 10 tys. USD miesięcznie. Nie chodzi oczywiście o to, aby porównywać płace w USA z polskimi, ale warto przyjrzeć się pewnym relacjom. Otóż w USA nawet najlepsze firmy i koncerny oferują wynagrodzenie wyższe tylko o kilkadziesiąt procent niż administracja, a nie kilkakrotnie wyższe, jak to jest u nas.

Na rynku pracy obowiązuje prosta zasada – im mniej płacisz, tym gorszych masz fachowców. Są oczywiście rzadkie wyjątki, związane ze statystyką zjawisk społecznych, jak też z oferowanymi dodatkowo apanażami, nieprzeliczalnymi bezpośrednio na pieniądze. Problem podwyższenia informatykom wynagrodzeń w administracji rządowej wynika z siatki płac, w której nie uwzględniono faktu, że specjaliści niektórych dziedzin (np. informatyka, prawo, ekonomia) mogą otrzymać wielokrotnie wyższe wynagrodzenie, zatrudniając się w firmach prywatnych lub zagranicznych koncernach. Z tego samego powodu szkoły cierpią na brak nauczycieli języka angielskiego i informatyki.

Nie zgłaszają się więc do administracji najlepsi absolwenci, których często już podczas studiów wiążą ze sobą zagraniczne firmy. Nikt nie widział jeszcze na uczelniach „łowców głów", pracujących dla administracji państwowej.

Jednocześnie odchodzą starsi i doświadczeni specjaliści, dla których prestiż instytucji o dumnie brzmiącej nazwie stracił już walor ważnego ekwiwalentu wynagrodzenia. W efekcie pozostaje w administracji bardzo niewielu profesjonalistów. „Duży procent niefachowców stworzy problem w naborze lepszych kadr, gdyby sektor publiczny zdecydował się na poprawę warunków płacowych swoich informatyków" – stwierdza dokument rządowy.

Prywatne przedsiębiorstwa, przede wszystkim banki i firmy ubezpieczeniowe, gdy brakuje im odpowiedniej kadry lub mają kłopoty z wyegzekwowaniem niezbędnych usług od swoich pracowników, zlecają pełną obsługę informatyczną wyspecjalizowanej firmie z zewnątrz (tzw. outsourcing). Administracja nie zawsze może sobie na takie rozwiązanie pozwolić, nawet gdyby było tańsze i lepsze. Są na przykład systemy, które powinna wykonać wyłącznie polska firma, a obsługiwać je mogą tylko ludzie związani tajemnicą służbową. Stawiając tylko te niezbyt wygórowane warunki Komenda Główna Policji nie może znaleźć wykonawcy sieciowego systemu, spełniającego wysokie wymagania niezawodności i bezpieczeństwa.

Poważne kłopoty, jakie wynikają z braku wysoko kwalifikowanych kadr informatycznych w administracji, powstają przy próbach nawiązania współpracy międzynarodowej, zwłaszcza z Unią Europejską.

dokończenie na s. 39 →

fot. Lesław Wawrzonek

WYDAWNICTWO SIGMA NOT

Cena 5,00 zł (50 000 zł)

informatyka

Miesięcznik
ISSN 0542-9951
INDEKS 36124

Uwaga

ukazuje się już elektroniczna wersja naszego miesięcznika. Oto nasz adres w Internecie:

http://www.pol.pl/informatyka/

nr 7 1996
Lipiec
Rok wydania XXXII

W numerze:

Strona

Rada w radę o pracownikach - *Lesław Wawrzonek* I okł.

publikacje

Komputerowe przetwarzanie języka naturalnego - *Małgorzata Marciniak, Anna Kupść, Agnieszka Mykowiecka* 1

Analiza specyfiki programowania obiektowego w środowisku implementacyjnym SQLWindows - *Jacek Szedel, Michał Kolano* 6

Koncepcja modelu dla obiektowej bazy danych - *Maciej Piasecki* 12

Model edukacji informatycznej społeczeństwa - *Marek Miłosz* 16

Zagadnienie reprezentacji czasu w klasycznych systemach baz danych - *Krzysztof Kania, Jerzy Gołuchowski* 20

Zasady licencjonowania i zakupu pakietów rodziny Microsoft BackOffice 24

Systemy czasu rzeczywistego

Metoda projektowania HOOD - *Bogumiła Hnatkowska, Zbigniew Huzar* 27

informacje

Nowe władze PTI - *Lesław Wawrzonek* 33

Szybkie wytwarzanie aplikacji na Wiosennej Szkole PTI - *Leszek Bogusławski, Zdzisław Szyjewski* 35

I Ogólnopolskie Zawody w Programowaniu Zespołowym 37

Sobota?, odpada. - *Leszek Bogusławski* 39

40

W najbliższych numerach:

- **Maciej Piasecki przedstawia OB!OK! - obiektowy system zarządzania bazą danych.**
- **Krzysztof Kania i Jerzy Gołuchowski omawiają problem reprezentacji czasu w modelu konceptualnym SI.**
- **Andrzej Pająk zastanawia się nad wyznacznikami kształcenia informatyków.**

REDAGUJE ZESPÓŁ:
dr Lesław **WAWRZONEK**
(redaktor naczelny)
mgr Krystyna
KARWICKA-RYCHLEWICZ
(z-ca redaktora naczelnego)
mgr Jerzy **SZYLLER**
(redaktor)
mgr Zdzisław **ŻURAKOWSKI**
(redaktor działu)
Agnieszka **PAPIS**
(sekretarz redakcji)
Alina **KLEPACZ**
(administracja redakcji)

KOLEGIUM REDAKCYJNE:
prof. dr hab. Leonard BOLC,
mgr inż. Piotr FUGLEWICZ,
prof. dr hab. Jan GOLIŃSKI,
dr inż. Zenon KULPA,
prof. dr inż. Jan MULAWKA,
prof. dr hab. Wojciech OLEJNICZAK,
mgr inż. Jan RYŻKO,
dr Witold STANISZKIS,
dr inż. Jacek STOCHLAK,
prof. dr hab. Maciej STOLARSKI,
prof. dr hab. Zdzisław SZYJEWSKI,
prof. dr hab. inż.
Ryszard TADEUSIEWICZ,
prof. dr hab. Jan WĘGLARZ

PRZEWODNICZĄCY
RADY PROGRAMOWEJ
prof. dr hab.
Juliusz Lech KULIKOWSKI

WYDAWCA:
Wydawnictwo Czasopism i Książek Technicznych SIGMA NOT Spółka z o.o.
ul. Ratuszowa 11
00-950 WARSZAWA
skrytka pocztowa 1004

Redakcja:
00-950 Warszawa,
ul. Ratuszowa 11, p. 628 i 644
skrytka pocztowa 1004
tel., fax: 619-11-61
tel.: 619-22-41 w. 159
e-mail:
informat@pol.pl

Materiałów nie zamówionych redakcja nie zwraca

Autorzy artykułów proszeni są o przysyłanie tekstów na dyskietkach $3^1/_2$" – w czystych kodach ASCII (Latin II, Mazovia) lub edytorach: Word 2, Word 6 – oraz odbitki na papierze. Teksty nie mogą przekraczać 20 tys. znaków.

Redakcja nie ingeruje w treść i formę ogłoszeń i innych materiałów reklamowych, w związku z tym nie ponosi za nie odpowiedzialności.

Ogłoszenia przyjmują:

- **Redakcja**
- **Dział Reklamy i Marketingu**
  00-950 Warszawa,
  ul. Mazowiecka 12
  tel.: 27-43-66,
  fax: 26-80-16

Warunki prenumeraty na 1996 r.

Zamówienia na prenumeratę czasopism wydawanych przez Wydawnictwo SIGMA-NOT można składać w dowolnym terminie. Mogą one obejmować dowolny okres czasu, tzn. dotyczyć dowolnej liczby kolejnych zeszytów każdego czasopisma.

Zamawiający może otrzymywać zaprenumerowany przez siebie tytuł począwszy od następnego miesiąca po dokonaniu wpłaty. Zamówienia na zeszyty sprzed daty otrzymania wpłaty będą realizowane w miarę możliwości – z posiadanych zapasów magazynowych.

Warunkiem przyjęcia i realizacji zamówienia jest otrzymanie z banku potwierdzenia dokonania wpłaty przez prenumeratora. Dokument wpłaty jest równoznaczny ze złożeniem zamówienia.

Wpłat na prenumeratę można dokonywać na ogólnie dostępnych blankietach w Urzędach Pocztowych (przekazy pieniężne) lub Bankach (polecenie przelewu), przekazując środki pod adres:
Wydawnictwo SIGMA-NOT Spółka z o.o.
Zakład Kolportażu
00-716 Warszawa, skr. poczt. 1004
konto:
PBK S.A. III 0/Warszawa nr 370015-1573-2720-3-67

Wpłaty na prenumeratę przyjmują także wszystkie urzędy pocztowe nadawczo-odbiorcze oraz doręczyciele na terenie całego kraju

Na blankiecie wpłaty należy czytelnie podać nazwę zamawianego czasopisma, liczbę zamawianych egzemplarzy, okres prenumeraty oraz własny adres.
Na życzenie prenumeratora, zgłoszone np. telefonicznie, Zakład Kolportażu ul. Bartycka 20, 00-950 Warszawa, (telefony: 40-30-86, 40-35-89 oraz 40-00-21 wew. 249, 293, 299) wysyła specjalne blankiety zamówień wraz z aktualną listą tytułów i cennikiem czasopism.

Odbiorcy zagraniczni mogą otrzymywać czasopisma poprzez prenumeratę dewizową (wpłata dokonywana poza granicami Polski w dewizach, wg cennika dewizowego z cenami podanymi w dolarach amerykańskich) lub poprzez zamówioną w kraju prenumeratę ze zleceniem wysyłki za granicę (zamawiający podaje dokładny adres odbiorcy za granicą, dokonując równocześnie wpłaty w wysokości dwukrotnie wyższej niż cena normalnej prenumeraty krajowej).

Egzemplarze archiwalne (sprzedaż przelewowa lub za zaliczeniem pocztowym) można zamawiać pisemnie, kierując zamówienia pod adresem: Wydawnictwo SIGMA NOT, Spółka z o.o. Zakład Kolportażu, 00-716 Warszawa, ul. Bartycka 20, paw. B, tel. 40-37-31, natomiast za gotówkę można je nabyć w Klubie Prasy Technicznej w Warszawie ul. Mazowieckiej 12, tel. 26-80-17.

Istnieje możliwość zaprenumerowania 1 egz. czasopisma po cenie ulgowej przez indywidualnych członków stowarzyszeń naukowo-technicznych zrzeszonych w FSNT oraz przez uczniów zawodowych i studentów szkół wyższych. Blankiet wpłaty na prenumeratę ulgową musi być opatrzony na wszystkich odcinkach pieczęcią koła SNT lub szkoły.

W przypadku zmiany cen w okresie objętym prenumeratą Wydawnictwo zastrzega sobie prawo do wystąpienia o dopłatę różnicy cen oraz prawo do realizowania prenumeraty tylko w pełni opłaconej.
Cena jednego egzemplarza: normalna 5,00 zł (50 000 zł), ulgowa 3,75 zł (37 500 zł)
Wartość prenumeraty w zł:
Normalna: kwartalna 15,00 zł (150 000 zł), półroczna 30,00 zł (300 000 zł), roczna 60,00 zł (600 000 zł)
Ulgowa: kwartalna 11,25 zł (112 500 zł), półroczna 22,50 zł (225 000 zł), roczna 45,00 zł (450 000 zł)

Skład i druk: Drukarnia SIGMA NOT Sp. z o.o. z. 233/96

publikacje

Komputerowe przetwarzanie języka naturalnego – wybrane zagadnienia

Anna Kupść, Małgorzata Marciniak, Agnieszka Mykowiecka
Instytut Podstaw Informatyki PAN
Warszawa

Najpopularniejszym sposobem porozumiewania się ludzi między sobą jest język naturalny. Niekwestionowana pozycja języka jest związana z tym, że stanowi on bardzo uniwersalny i efektywny środek wyrazu, pozwalający na przekazywanie różnorodnych informacji w sposób zrozumiały dla wielu odbiorców. Ponieważ język jest wykorzystywany nie tylko do bieżącej wymiany informacji, ale także do zapisu wiedzy, istnieją i ciągle są rozszerzane ogromne zbiory tekstowe. Z uwagi na wielkość tych zasobów dostęp do zawartych w nich danych staje się coraz bardziej utrudniony. Istnieją więc co najmniej dwa podstawowe powody do podjęcia prac nad próbami komputerowego przetwarzania wypowiedzi w języku naturalnym.

Powszechność języka naturalnego jako środka porozumiewania się pozwala przypuszczać, że mógłby on stanowić dobry sposób porozumiewania się ludzi z systemami komputerowymi, z drugiej strony jest widoczna potrzeba zautomatyzowania procesu przetwarzania danych tekstowych.

Wśród zastosowań, które są powiązane z komputerowym przetwarzaniem języka naturalnego, można wyróżnić następujące zagadnienia:

- umożliwienie dialogu z systemami komputerowymi w języku naturalnym,
- automatyczne przetwarzanie tekstów (wyszukiwanie informacji, indeksacja, tworzenie streszczeń),
- wspomaganie tłumaczenia,
- tworzenie tekstów w języku naturalnym (automatyczne generowanie opisów zdjęć, raportów, listów informacyjnych, opatrywanie komentarzem różnego typu analiz formalnych, np. statystycznych czy finansowych),
- analiza i synteza mowy,
- wspomaganie nauki języka obcego (generowanie ćwiczeń, analiza błędów),
- wspomaganie redagowania tekstów (korektory pisowni i składni),
- tworzenie i utrzymywanie danych o języku (słowniki, tezaurusy),
- wspomaganie badań lingwistycznych (analiza dużych zbiorów tekstów, weryfikacja hipotez).

Analiza i synteza mowy wymaga zarówno wiedzy na temat przetwarzania sygnału, jak i wiedzy informatycznej. W ramach tej dziedziny są prowadzone prace nad udoskonaleniem urządzeń generujących lub rejestrujących sygnały oraz nad oprogramowaniem służącym do jego obróbki. Istniejące systemy coraz lepiej radzą sobie z generowaniem dźwięków ludzkiej mowy, a także z ich rozpoznawaniem i przetwarzaniem na znakowy zapis tekstu, który może podlegać dalszej obróbce komputerowej.

Problemy związane z przetwarzaniem tekstów pisanych

Inne problemy dotyczą analizy i syntezy tekstów pisanych. Aby tego rodzaju zadanie było wykonalne, trzeba dysponować bardzo różnorodną wiedzą z możliwością sprawnego do niej dostępu. Na ogół niezbędne jest bowiem korzystanie z:

- wiedzy morfologicznej dotyczącej budowy i sposobu odmiany wyrazów oraz pozwalającej na ich identyfikację,
- wiedzy syntaktycznej opisującej sposób konstruowania poprawnych zdań,
- wiedzy semantycznej zawierającej informacje o znaczeniu poszczególnych słów oraz sposobie ich łączenia w poprawne znaczeniowo zdania,
- wiedzy pragmatycznej dotyczącej sposobu używania zwrotów i zdań w różnych kontekstach oraz wpływu kontekstu na znaczenie wypowiedzi,
- wiedzy o świecie niezbędnej do prawidłowego interpretowania informacji.

Kolejną ważną grupą zagadnień przetwarzania języka naturalnego jest oprogramowanie ogromnych baz da-

nych, jakimi są komputerowe słowniki oraz tezaurusy, w których gromadzona jest wiedza dotycząca określonej dziedziny.

Analiza morfologiczna

Zadaniem analizy morfologicznej jest rozpoznawanie poszczególnych słów tekstu oraz kojarzenie z nimi wszelkich informacji lingwistycznych niezbędnych do jego dalszego przetwarzania. Niestety, na tym etapie analizy najczęściej nie jest możliwe ustalenie jednoznacznej informacji leksykalnej skojarzonej z daną formą wyrazową. Należy więc uwzględniać wszystkie możliwe warianty dotyczące tak informacji leksykalnej, jak i znaczenia. Przykładowo rzeczownik *drzewa* może reprezentować zarówno dopełniacz liczby pojedynczej, jak też mianownik lub biernik liczby mnogiej.

Istnieje wiele form wyrazowych, które mogą się wywodzić od słów mających różne znaczenia, mogą one też reprezentować różne części mowy. Rozważmy wystąpienie słowa *śledzi* w następujących zdaniach:

(1) *Janek **śledzi** rozgrywki piłkarskie.*
(2) *Janek nie kupił **śledzi**.*

W powyższym przykładzie niejednoznaczność może być rozstrzygnięta na etapie analizy syntaktycznej. Słowo *śledzi* ma bowiem jasno określone funkcje gramatyczne. Ustalenie sensu niektórych słów wymaga nie tylko odwoływania się do kontekstu wynikającego z przyległych zdań, ale również do wiedzy o świecie. Oto przykład takiego zdania:

(3) *Janek z zainteresowaniem oglądał zdjęcia **gwiazd**.*

Na podstawie powyższego, wyrwanego z kontekstu zdania nie jesteśmy w stanie ustalić, czy Janek oglądał zdjęcia artystów czy też ciał niebieskich. Jeśli dodatkowo wiemy, że Janek jest astronomem, to z dużym prawdopodobieństwem możemy wnioskować, że chodzi tu o ciała niebieskie. Jeśli natomiast Janek interesuje się filmem, to skłaniamy się raczej do interpretacji słowa *gwiazd* jako artystów. W takim wypadku do jednoznacznego ustalenia znaczenia słowa dojdzie dopiero na etapie analizy semantycznej, uwzględniającej elementy wnioskowania.

Analiza syntaktyczna

Zadaniem analizy syntaktycznej jest rozstrzygnięcie, czy dane zdanie jest poprawne gramatycznie oraz dokonanie jego rozbioru. Na tym etapie również pojawiają się problemy z niejednoznacznością. Większość zdań ma kilka poprawnych rozbiorów w ramach danej gramatyki. Standardowym przykładem jest tu następujące zdanie angielskie:

(4) I saw a man in the park with a telescope.
Ja widziałem człowieka w parku przez/z teleskop/em.

Zdanie to może mieć kilka poprawnych rozbiorów gramatycznych, gdyż w parku może się znajdować obserwator lub osoba obserwująca, a teleskop może być zarówno narzędziem, przez które prowadzona jest obserwacja, jak i rekwizytem posiadanym przez osobę obserwowaną.

Niejednokrotnie poprawny rozbiór gramatyczny może się okazać pozbawiony sensu. Oto przykład:

(5) *Lekarstwo podano choremu **w postaci zastrzyku**.*
(6) *Lekarstwo podano choremu **w stanie śpiączki**.*

Obydwa zdania są tak samo skonstruowane, aczkolwiek ich semantycznie poprawne rozbiory syntaktyczne różnią się. W zdaniu (5) podkreślona przydawka jest sensownym określeniem lekarstwa, a nie chorego. W zdaniu (6) sytuacja jest odwrotna. Wybór właściwych rozbiorów składniowych nie jest więc czasami możliwy bez głębszej analizy sensu zdania.

Innym istotnym zagadnieniem jest problem elips. Występują one w zdaniach niekompletnych, w których pominięto pewne ważne części składniowe. Zjawisko to wymaga wnikliwego potraktowania, szczególnie w wypadku analizy dialogów, w których pojawia się bardzo często. Oto przykład:

(7) *A: Ile kosztowała książka?*
B: Czternaście złotych.
A: A gazety?
B: Dwa.

Powyższy dialog jest dla człowieka w pełni zrozumiały, pomimo że jedynym kompletnym zdaniem jest pierwsze pytanie. Pojawiają się tu dwa problemy: kiedy niepełne zdanie możemy uznać za poprawne syntaktycznie oraz jak należy je uzupełnić brakującymi frazami.

Analiza semantyczna

Zadaniem analizy semantycznej jest ustalenie znaczenia zdania, czyli przetłumaczenie go na określoną postać logiczną. Jak już pokazaliśmy, niektóre wcześniej opisane problemy mogą być rozwiązane dopiero przy użyciu pewnych mechanizmów wnioskowania umożliwiających ustalenie, czy dana formuła logiczna nie jest sprzeczna z wiedzą posiadaną na dany temat.

Bardzo istotną częścią analizy semantycznej są problemy dotyczące odwołań (referencji), czyli ustalanie zależności między frazami opisującymi pewne obiekty, a obiektami świata rzeczywistego. Jednym z elementów tego zagadnienia jest teoria wiązania, która na podstawie konstrukcji syntaktycznej zdania ustala zasady odpowiedzialne za interpretację zaimków, czyli opisuje ich dopuszczalne powiązania z frazami rzeczownikowymi. Rozważmy następujące zdanie:

(8) *Jan powiedział Piotrowi, że **go** widział.*

W zdaniu (8) podkreślony zaimek może być powiązany z Piotrem lub inną, nie wymienioną w zdaniu osobą,

nie może jednak odnosić się do Jana. Ostateczna interpretacja tego zdania wymaga ustalenia do kogo naprawdę odnosi się ten zaimek. Problem odwołań jest bardzo złożony. Możliwe są bowiem odniesienia nie tylko do całych opisywanych obiektów, ale, przykładowo, w wypadku zbiorów, do jego elementów lub fragmentów pewnej całości.

(9) *Komputer jest zepsuty. Trzeba wymienić zasilacz.*

Właściwe odczytanie przykładu (9) wymaga ustalenia, że należy wymienić zasilacz konkretnego zepsutego komputera (tego wymienionego w poprzednim zdaniu), a nie dowolny inny.

Przedstawione tu zagadnienia stanowią tylko niewielką część problemów, które pojawiają się w procesie analizy tekstu. Zależało nam jedynie na uświadomieniu Czytelnikowi, że problem analizy tekstu jest zagadnieniem niezwykle złożonym, pomimo że każde dziecko z łatwością prawidłowo odczytuje znaczenie przytoczonych tu przykładów.

Generowanie tekstów

Komunikacja w języku naturalnym jest dialogiem, tak więc poza analizą wypowiedzi istnieje też problem syntezy tekstów. W procesie tym wyróżnić można następujące etapy: ustalenie treści wypowiedzi, opracowanie sposobu przekazania wybranych faktów oraz ostateczne sformułowanie poszczególnych zdań. Sposób wyboru faktów, które należy uwzględnić w tekście, zależy od konkretnego zastosowania. Przykładowo, w wypadku opisu zdjęć zestaw faktów jest wyznaczony przez sposób reprezentacji treści zdjęcia. Natomiast przy udzielaniu odpowiedzi na pytania trzeba dokonać selekcji z bardzo rozbudowanej bazy wiedzy. Najtrudniejszym problemem związanym z generowaniem tekstów jest ustalenie jego formy. Uporządkowanie wyselekcjonowanych faktów, wybór sposobu ich przedstawienia czy nadanie całemu tekstowi określonego zabarwienia to tylko niektóre z bardzo licznych kwestii wymagających rozstrzygnięcia. W tym wypadku trudno jest nie tyle wskazać na możliwe warianty, co ustalić zależność dokonywanych wyborów od różnych cech związanych z kontekstem wypowiedzi. Najbardziej znane rozwiązania tego problemu to przyjęcie zbioru schematów możliwych tekstów [9] lub planowanie postaci wypowiedzi na podstawie zestawu relacji retorycznych, dla których zdefiniowane są cele, jakie przy ich użyciu można osiągnąć (*Rethorical Structure Theory* [8]).

Ostatni etap generowania tekstu, czyli ustalenie ostatecznej formy wypowiedzi, jest do pewnego stopnia odwróceniem omawianego wcześniej procesu rozpoznawania struktury syntaktycznej zdań. Wykorzystywane są tu te same metody opisu języka, choć tylko w niektórych wypadkach udało się zastosować tę samą gramatykę, zarówno na potrzeby analizy, jak i syntezy wypowiedzi.

Dla zilustrowania wymienionych problemów załóżmy, że musimy wyrazić w języku polskim następujący zbiór faktów (użyta tu notacja jest bardzo uproszczona):

(10) Piotr(x), książka(y), notatka(z),
szukać(x,z,t1), podnieść(x,y,t2), zobaczyć(x,z,t3),
t1 < t2 < t3 < TERAZ

Nawet tak prosty zestaw informacji można wyrazić na wiele różnych sposobów, a wybór konkretnego wariantu zależy od bardzo szeroko rozumianego kontekstu. Oto niektóre z możliwych interpretacji:

(11) *Piotr szukał notatki. Piotr podniósł książkę. Piotr zobaczył notatkę.*

(12) *Piotr szukał notatki. Zobaczył ją jak tylko podniósł książkę.*

(13) *Po podniesieniu książki Piotr zobaczył notatkę, której szukał.*

(14) *Piotr podniósł książkę i wtedy zobaczył notatkę, której szukał.*

(15) *Notatkę, której szukał, Piotr zobaczył dopiero po podniesieniu książki.*

Powyższy przykład ilustruje, jaki wpływ na postać generowanego tekstu mają decyzje dotyczące kolejności przedstawiania faktów, grupowania ich w zdania, sposobu łączenia zdań pojedynczych w złożone czy miejsca użycia zaimków. Warto przy tym zauważyć, że proste połączenie poprawnych interpretacji poszczególnych faktów na ogół nie prowadzi do zadowalającego efektu (tak jak w (11)).

Formalne metody opisu języka naturalnego

Rozwiązanie problemów zasygnalizowanych w poprzedniej części artykułu wymaga stworzenia formalnego opisu języka naturalnego. Tradycyjne gramatyki nie nadają się do wykorzystania w systemach komputerowych, gdyż przedstawiają zasady budowy wypowiedzi w języku naturalnym w sposób opisowy, a poza tym wyrywkowy i niepełny. Na podstawie zawartych w nich informacji można (i to też nie zawsze) stwierdzić, czy konkretne wyrażenie jest zgodne z normą danego języka, ale nie można podać metody konstruowania poprawnych fraz. Dopiero zapoczątkowany przez Chomsky'ego [3] rozwój gramatyk formalnych, dostosowanych do specyfiki języków naturalnych, doprowadził do powstania nowoczesnych metod ich opisu. W chwili obecnej istnieje wiele konkurencyjnych metod zapisu struktury tekstów. Niestety, język naturalny jest tak bogaty, że w żadnym z opracowanych do tej pory formalizmów nie udało się opisać wszystkich występujących w nim zjawisk, a wśród zajmujących się tą tematyką badaczy panuje brak zgody co do tego, która z istniejących metod jest najodpowiedniejsza.

Budowa syntaktyczna zdań jest przedstawiana zazwyczaj jako drzewo odzwierciedlające pewien wybrany typ związków między poszczególnymi elementami wypowiedzi. Na ogół za podstawę do analizy wybiera się strukturę frazową zdania, zgodnie z którą w zdaniu prostym (S) wyróżnić można frazę podmiotu, będącą frazą rzeczownikową (NP) oraz czasownikową frazę orzeczenia (VP). Fraza czasownikowa to – na przykład – czasownik nieprzechodni (V) lub czasownik przechodni wraz z frazą rzeczownikową odpowiadającą dopełnieniu. Z kolei prosta fraza rzeczownikowa może składać się z rzeczownika własnego (PN) lub pospolitego. Wymie-

nione zasady zapisać można w postaci odpowiednich reguł gramatycznych, np. (16).

(16) S ® NP VP

Jeżeli reguły te uzupełnimy wyliczając słowa należące do odpowiednich kategorii, to będziemy mogli za ich pomocą wygenerować nastepujące zdania.

(17) **Filip lubić Magda*

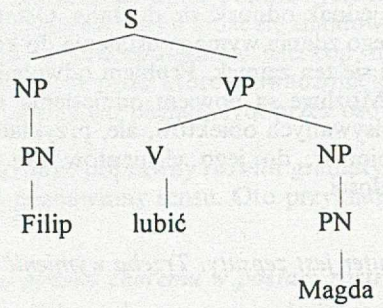
(18) **Magda dawać*

Aby nadać tym zdaniom poprawną postać uwzględniającą właściwe formy fleksyjne, trzeba rozbudować podane reguły gramatyczne o informacje dotyczące liczby, rodzaju i przypadku. Natomiast w celu wyeliminowania niepoprawnego przykładu (18) zbiór czasowników trzeba podzielić na takie, które występują samodzielnie i takie, które wymagają dopełnienia. Można to uczynić wprowadzając dodatkową dwuwartościową cechę *Trans*. Te zmiany nie gwarantują jeszcze poprawności generowanych zdań, nie jest – na przykład – uwzględniony fakt wymagania przez czasowniki dopełnień w określonych przypadkach (problem ten można rozwiązać przez podział czasowników na kilka grup). Pominięto też problem form fleksyjnych wyrazów, które mogą być albo przechowywane bezpośrednio w odpowiednim słowniku (co bardzo zwiększa jego objętość) albo syntetyzowane i rozpoznawane za pomocą odpowiednich algorytmów (ich stopień komplikacji jest różny dla różnych języków naturalnych).

Jak widać już na przykładzie bardzo prostych zdań, skonstruowanie reguł, które będą w czytelny sposób opisywać zasady budowy wyrażeń w języku naturalnym, nie jest zadaniem łatwym. Prowadzone w tym zakresie badania mają na celu opracowanie metody opisu, która spełniałaby wiele kryteriów, z których najważniejsze to: uniwersalność, precyzyjność, zwięzłość, efektywność i czytelność. Wśród licznych istniejących obecnie formalizmów na uwagę zasługują między innymi LFG (*Lexical-Functional Grammar*, [2]), uogólniona gramatyka struktur frazowych (GPSG, [4]) oraz HPSG (*Head-driven Phrase Structure Grammar*, [10], [11]).

W procesie przetwarzania języka naturalnego opis struktury syntaktycznej wypowiedzi nie jest celem samym w sobie, ale stanowi etap pośredni na drodze do ustalenia jej znaczenia. Przykładowymi metodami reprezentacji semantyki wyrażeń języka naturalnego są: semantyka sytuacyjna (*Situational Semantics* [1]) i DRT (*Discourse Representation Theory* [6]). Zaproponowana przez Kampa i Reyle metoda reprezentacji znaczenia wypowiedzi polega na odwzorowywaniu jej znaczenia w postaci specjalnej struktury, nazwanej DRS. Struktura ta składa się z dwóch części: zbioru obiektów (ang. *discourse referents*), zwanych uniwersum oraz zbioru warunków opisujących zależności między tymi obiektami. Analiza wypowiedzi w teorii DRT rozpoczyna się od utworzenia struktury DRS zawierającej drzewo budowy syntaktycznej. Struktura ta jest następnie przekształcana za pomocą ustalonego zbioru reguł. Przykładowo analizę syntaktyczną zdania (17), reprezentowaną przez drzewo wywodu (19), można zapisać w postaci struktury (20).

(19)



(20)

x y
Filip (x)
Magda(y)
[x lubić y]

Do uzyskanej w ten sposób struktury DRS można dołączyć opis struktury syntaktycznej kolejnego zdania. Cecha ta w naturalny sposób pozwala na wykorzystanie DRT do opisu znaczenia dłuższych fragmentów tekstu. Struktury DRS wyrazić można za pomocą formuł logiki predykatów pierwszego rzędu i korzystać z odpowiednich dla tej logiki metod wnioskowania.

Niestety nie wszystkie treści zawarte w wypowiedziach w języku naturalnym jesteśmy w stanie zapisać w sposób formalny, a żadna z istniejących logik nie pozwala na pełną symulację rozumowania, przeprowadzanego przez ludzi posługujących się językiem naturalnym. Badania w tej dziedzinie prowadzą jednak do stopniowego rozszerzania zakresu reprezentowanych znaczeń i tym samym pozwalają na zwiększanie liczby możliwych zastosowań metod automatycznej analizy i syntezy tekstów w języku naturalnym.

Praktyka – aktualne możliwości

W wyniku dynamicznego rozwoju lingwistyki komputerowej (powstawanie nowych teorii lingwistycznych, gramatyk formalnych i systemów do ich implementacji) oraz dążenia do ułatwienia pracy z komputerami powstają liczne programy i systemy użytkowe wykorzystujące języki naturalne. Dotyczy to przede wszystkim takich języków, jak: angielski, niemiecki, japoński czy francuski. Najbardziej popularnymi zastosowaniami są słowniki komputerowe, programy wspomagające redagowanie tekstów, poprawiające pisownię, a także programy sprawdzające (w pewnym zakresie) formę syntaktyczną zdań. Kolejnym bardzo ważnym zastosowaniem jest wspomaganie tłumaczenia tekstów (całkowicie automatyczne tłumaczenie tekstów nie jest na razie możliwe i zapewne długo jeszcze nie będzie). Przykładem takiego systemu tłumaczącego jest kanadyjski system METEO. Służy on do dwukierunkowego tłumaczenia prognozy pogody z angielskiego na francuski. Innym przykładem jest niemiecki projekt badawczy VERBMOBIL, obejmujący zagadnienia związane z planowaniem spotkań. Istnieją także systemy wspomagające tłumaczenie tekstów technicznych czy specjalistycznej dokumentacji. Ich podstawową rolą jest wykonanie żmudnej pracy szukania odpowiednich terminów specjalistycznych czy określonych zwrotów i dostarczenie wstępnej wersji tekstu, np. europejski system SYSTRANS, wspomagający tłumaczenie dokumentów na potrzeby instytucji Unii Europejskiej.

Wiele programów wykorzystuje język naturalny do udzielania odpowiedzi czy informowania użytkownika o popełnionych błędach. Komunikacja ta jest jednak daleka od prawdziwego dialogu w języku naturalnym i polega w zdecydowanej większości przypadków na wykorzystywaniu gotowych wzorców (tekstu lub cyfrowego zapisu dźwięku). Wzorce te mogą być uzupełnianie konkretnymi, dobranymi do sytuacji słowami. Przykładem takiego programu jest system VoiceReport wykorzystywany do sporządzania raportów medycznych w szpitalach holenderskich (komunikacja z nim odbywa się w języku angielskim). Całkowicie zautomatyzowane generowanie tekstów jest dokonywane na razie przez programy o charakterze eksperymentalnym, których zastosowanie ogranicza się bądź wyłącznie do testów, bądź do pewnej konkretnej sytuacji czy grupy odbiorców. Jako przykład takich zastosowań testowych podać można system DIAMOND [5], który udziela wyjaśnień dotyczących rozwiązań proponowanych przez system OFFICE-PLAN, planujący rozmieszczenie pracowników biura. Innym przykładem jest system analizujący informacje dotyczące atrakcji turystycznych Düsseldorfu [7], który powstał w ramach projektu LILOG (wspólnego przedsięwzięcia IBM i kilku uniwersytetów niemieckich).

Stopień zaawansowania prac prowadzonych nad komputerowym przetwarzaniem języka polskiego nie dorównuje badaniom podjętym w tym zakresie dla języków, takich jak angielski, niemiecki, japoński czy francuski. Przyczyną tego stanu rzeczy jest przede wszystkim niewielkie zainteresowanie problemami automatycznego przetwarzania języka naturalnego w naszym kraju. Dotychczasowe prace w tej dziedzinie były ograniczone głównie do morfologii, akustycznej analizy i syntezy mowy oraz tworzenia słownikowych baz danych. Wśród nielicznych opracowanych programów można wymienić korektory ortografii firm: TiP, Malkom, Litterae i Morfologic, oparte na dość bogatych słownikach morfologicznych oraz analizator morfologiczny dla ponad 100 tys. wyrazów, który powstał w Instytucie Informatyki UW. Niedługo ma się ukazać elektroniczna wersja PWN-owskiego słownika języka polskiego również dysponująca analizatorem morfologicznym. Słownik fleksyjny powstaje także w Instytucie Języka Polskiego PAN w Krakowie. Istniejąca aktualnie wersja to zawierający ok. 15 tys. haseł *Słownik wyrazów trudnych*. Praktyczne zastosowanie znalazły polskie syntezatory mowy; są też prowadzone badania nad prototypem systemu rozpoznającego mowę.

* * *

Zagadnienia przedstawione w niniejszym artykule nie wyczerpują problemów związanych z komputerowym przetwarzaniem języków naturalnych. Języki te (nawet angielski) nadal nie są do końca zbadane i opisane, a metody, jakimi dysponuje obecnie inżynieria lingwistyczna, pozwalają na zapis tylko niektórych z ustalonych już prawidłowości. Tym niemniej nawet ta cząstkowa wiedza pozwala na konstruowanie wielu użytecznych programów o stale wzrastającym zakresie zastosowania. Wielkość i dostępność danych dotyczących języka polskiego oraz liczba dostosowanych do ich przetwarzania narzędzi programistycznych zdecydowanie ustępuje odpowiednim zasobom dla większości języków zachodnioeuropejskich, a nawet języków takich jak czeski. Jednak zainteresowanie omawianymi tu problemami powoli wzrasta, co być może pozwoli na włączenie języka polskiego do międzynarodowych prac dotyczących inżynierii lingwistycznej.

LITERATURA

[1] Barwise J., Perry J.: Situations and Attitudes. The MIT Press, Cambridge, Mass., 1983

[2] Bresnan J.: The passive in lexical theory. W Joan Bresnan (red.), The Mental Representation of Grammatical Relations, MIT Press Series on Cognitive Theory and Mental Representation, The MIT Press, Cambridge, Mass., 1982

[3] Chomsky N.: Syntactic Structures. Mouton, The Hague, 1957

[4] Gazdar G., Klein E., Pullum G. K. i Sag I. A.: Generalized Phrase Structure Grammar. Oxford: Basil Blackwell; Cambridge, Mass.: Harvard University Press, 1985

[5] Horacek H.: Decision making throughout the generation process in the systems WISBER and DIAMOND. W H. Horacek i M. Zock (red.), New Concepts in Natural Language Generation. Pinter Publishers, London, 1993

[6] Kamp H., Reyle U.: From Discourse to Logic. Kluwer Academic Publishers, Dordrecht, 1993

[7] Krifka-Dobes Z., Nowak H.-J.: From constituent planning to text planning. W M.Zock H.Horacek (red.), New Concepts in Natural Language Generation. Pinter Publishers, London, 1993

[8] Mann W.C.: Text generation: The problem of text structure. W D. D. McDonald i L. Bolc (red.), Natural Language Generation Systems. Springer-Verlag, New York, 1988

[9] McKeown K.: Discourse strategies for generating natural language TEXT. Artificial Intelligence, 27, 1985

[10] Pollard C., Sag I. A.: Information-Based Syntax and Semantics, Volume 1: Fundamentals. CSLI, 1987

[11] Pollard C., Sag I. A.: Head-driven Phrase Structure Grammar. Chicago University Press, 1994

[12] Świdziński M.: Gramatyka formalna języka polskiego, tom 349, Rozprawy Uniwersytetu Warszawskiego. Wydawnictwa UW, Warszawa, 1992

[13] Szpakowicz S.: Formalny opis składniowy zdań polskich. Wydawnictwa UW, Warszawa, 1986.

Richard Barker, Cliff Longman: CASE* Method SM - Modelowanie funkcji i procesów. Z angielskiego przełożyła: dr Małgorzata Szadkowska-Rucińska. WNT, Warszawa 1996, wyd.I, s.435, cena 48,0 zł, ISBN 83-204-1954-9

Książka ta stanowi nowoczesny poradnik projektowania systemów informacyjnych. Autorzy pracowali razem przez prawie dziesięć lat nad metodami CASE, studiami strategii oraz opracowaniem narzędzi CASE.

Projektowanie systemów informacyjnych leży na pograniczu dwóch dziedzin: zarządzania i informatyki. W dobie innowacji technologicznych i komunikacyjnych przedsiębiorstwa muszą rozwijać odpowiednie systemy wspierające. Temat ten został przedstawiony z wykorzystaniem metodyki CASE*Method, proponowanej przez firmę Oracle. Omówiono cel i zalety modelowania działania przedsiębiorstw. Szeroko przedstawiona została konstrukcja systemu czasu rzeczywistego i projekt implementacji systemu docelowego. Wszystkie te zagadnienia zostały wyjaśnione na podstawie przykładu z praktyki projektowania systemów, posługując się metodyką CASE*Method.

Książka jest przeznaczona przede wszystkim dla informatyków, analityków i projektantów systemów informacyjnych. Może też zainteresować osoby, zajmujące się zarządzaniem.**(a)**

Analiza specyfiki programowania obiektowego w środowisku implementacyjnym SQLWindows

Jacek Szedel
Michał Kolano
Zakład Oprogramowania
Instytut Informatyki
Politechnika Śląska
Gliwice

Artykuł rekomendowany

Niniejsze opracowanie poświęcono możliwościom programowania zorientowanego obiektowo z wykorzystaniem języka 4GL SQLWindows. Na wstępie w ogólny sposób przedstawiono podstawowe zagadnienia związane z programowaniem obiektowym. W części drugiej omówione zagadnienia odniesiono do języka programowania SQLWindows, wskazując na ich niekonwencjonalną realizację. Opisano także ograniczenia, jakie napotyka programista przy obiektowym tworzeniu aplikacji w tym środowisku. Przeprowadzona analiza dotyczy takich aspektów programowania obiektowego, jak budowa klas i obiektów, dziedziczenie, hermetyzacja, polimorfizm.

Pojawienie się koncepcji programowania zorientowanego obiektowo (*OOP*, ang. *Object Oriented Programming*) było niewątpliwie istotną zmianą w dziedzinie tworzenia oprogramowania. Jako pierwsze w mechanizmy OOP zostały wyposażone języki trzeciej generacji (*3GL*, ang. *third generation languages*). Z upływem czasu mechanizmy te ewoluowały, a programowanie obiektowe zyskiwało sobie coraz większą popularność. W ostatnich latach zaczęły pojawiać się narzędzia posługujące się językami czwartej generacji (*4GL* od ang. *fourth generation languages*). Języki te zostały stworzone w celu przyspieszenia procesu tworzenia aplikacji, w szczególności w dziedzinie zastosowań związanych z bazami danych. Początkowo nie dawały one żadnych możliwości programowania obiektowego lub czyniły to w sposób bardzo ograniczony. Znaczna popularność technik OOP spowodowała jednak, że kolejne wersje znanych języków 4GL zaczęto rozbudowywać w taki sposób, aby umożliwić obiektowe tworzenie oprogramowania z ich wykorzystaniem. Rozszerzenia obiektowe języków 4GL nie są jednak prostym przełożeniem mechanizmów OOP znanych z języków 3GL. W wielu przypadkach składnię języka poszerzono o elementy ściśle związane z systemem operacyjnym lub graficznym interfejsem użytkownika, zorientowanym na zdarzenia.

Przykładem takiego języka 4GL jest *SQLWindows*, wbudowany w środowisko projektowe GUPTA Corporate Editon firmy GUPTA Corporation. Język ten jest jednym z pierwszych języków 4GL, w który wbudowano mechanizmy programowania obiektowego. Realizacja tych mechanizmów jest niekonwencjonalna i nowatorska. Niniejsze opracowanie poświęcono analizie podstawowych mechanizmów programowania obiektowego w kontekście ich realizacji w języku SQLWindows. Zamieszczone przykłady dotyczą wersji 5.0 tego środowiska.

Rekomenduje:
Dr hab. inż. Stanisław Kozielski
Profesor nadzwyczajny Politechniki Śląskiej
Kierownik Zakładu Teorii Informatyki
Prodziekan Wydziału Automatyki, Elektroniki i Informatyki.

Mechanizmy występujące w językach zorientowanych obiektowo

Dziedziczenie

Dziedziczenie w rozumieniu programowania obiektowego najogólniej można scharakteryzować jako mechanizm przekazywania cech składowych klasy bazowej klasie pochodnej. Możliwe jest dziedziczenie składowych jednej lub wielu klas bazowych (dziedziczenie jedno- lub wielobazowe). Dziedziczenie jest podstawowym sposobem wyrażania podobieństwa pomiędzy obiektami, z uwzględnieniem zasad generalizacji i specjalizacji. Oznacza to, że bardziej generalne cechy obiektów znajdują się wyżej w hierarchii klas, a kolejne klasy potomne są coraz bardziej wyspecjalizowane.

Hermetyzacja (enkapsulacja)

Hermetyzacja polega na stosowaniu ograniczeń związanych z dostępem do poszczególnych komponentów klasy. Klasa, będąca agregacją pól i metod (tzn. zachowań) stanowić powinna zamkniętą całość i jako taka zamykać w sobie całościowe rozwiązanie pojedynczego problemu [2]. Poprawne funkcjonowanie klasy zależy w dużej mierze od sposobu w jaki komunikuje się ona ze świa-

tem zewnętrznym. Chodzi o to, aby komunikacja ta odbywała się według ściśle zdefiniowanego interfejsu i żeby sama klasa decydowała o tym, w które jej elementy i w jaki sposób można ingerować. W praktyce oznacza to możliwość definiowania odpowiednich praw dostępu do komponentów klasy na poziomie składni języka oraz stosowania akcesorów - funkcji dostępu do składowych.

Polimorfizm

Przez polimorfizm rozumie się możliwość umieszczania odwołań do metod klas pochodnych w obrębie klasy bazowej, nawet jeśli klasa pochodna nie została jeszcze zdefiniowana. Pozwala to przyjąć, że pewne usługi zostaną zaimplementowane w przyszłości w klasach pochodnych, i można odwoływać się do nich w czasie projektowania klasy bazowej. W czasie działania programu, odwołania do metod są kierowane automatycznie do odpowiednich klas pochodnych.

Agregacja

Agregacja, czyli zawieranie obiektów, zachodzi wówczas, gdy jakiś obiekt jest składową innej klasy. Klasa może zawierać jeden obiekt innej klasy (agregacja 1:1) lub zawierać ich wiele (agregacja 1:N). Zawieranie 1:N jest w praktyce realizowane przez umieszczanie w klasie tablic obiektów lub odpowiednich struktur dynamicznych.

Definiowanie klas i tworzenie obiektów

Pomiędzy klasą i obiektem klasy istnieje zasadnicza różnica: klasa jest abstrakcyjnym opisem (definicją) obiektu, natomiast obiekt jest tworzony w pamięci na podstawie wzorca, jakim jest klasa. Języki obiektowe pozwalają na statyczne bądź dynamiczne tworzenie obiektów. Obiekt będzie utworzony statycznie wówczas, gdy został odpowiednio zadeklarowany w kodzie źródłowym programu. Dynamiczne tworzenie obiektów odbywa się w trakcie działania programu w rezultacie wykonania odpowiednich operacji.

Składowe klasowe i indywidualne

Języki programowania zorientowane obiektowo w większości pozwalają na rozróżnienie w klasach składowych klasowych i indywidualnych. Składowa klasowa jest wspólna dla wszystkich wystąpień obiektów danej klasy. Istnieje fizycznie nawet wtedy, gdy nie został jeszcze utworzony żaden obiekt danej klasy. Składowe indywidualne są powielane dla każdego obiektu klasy i powstają dynamicznie w momencie tworzenia obiektu. Składowe klasowe i indywidualne określa się także czasami jako statyczne i dynamiczne.

Przeznaczenie podstawowych metod

Generalnie metody klasowe można podzielić na trzy podstawowe grupy [1]:

- konstruktory i destruktory - służące do inicjacji i deinicjacji obiektów,
- akcesory - pozwalające na dostęp do komponentów klasy (grupa metod związana ze stosowaniem hermetyzacji),
- realizatory - stanowiące implementację właściwych usług realizowanych przez klasę.

Programowanie obiektowe w SQLWindows

SQLWindows firmy GUPTA Corporation jest środowiskiem projektowym, służącym do tworzenia aplikacji z dziedziny baz danych, wykorzystujących graficzny interfejs użytkownika . Środowisko to zostało zaprojektowane i zoptymalizowane do współpracy z systemami baz danych wykorzystującymi model klient-serwer. Programy stworzone w SQLWindows potrafią komunikować się z serwerami baz danych pracującymi na różnych platformach sprzętowych i w różnych systemach operacyjnych [5].

SQLWindows jest strukturalnym językiem czwartej generacji, który od wersji 4.0 został rozbudowany o elementy OOP. Sposób realizacji tych mechanizmów jest próbą przystosowania klasycznego programowania obiektowego (znanego np. z C++) do nowoczesnego, zorientowanego na zdarzenia graficznego systemu operacyjnego.

Tworzenie aplikacji może odbywać się w sposób wizualny (ang. *visual programming*), co ułatwia szybkie przygotowanie gotowej, działającej aplikacji z pełnym interfejsem zewnętrznym. Atrybuty obiektów umieszczanych na ekranie można modyfikować podczas projektowania, dzięki czemu proces konstrukcji interfejsu bardzo się upraszcza. Ważną zaletą SQLWindows jest możliwość kodowania programu również w sposób klasyczny, tj. przez wprowadzanie jego tekstu z klawiatury w specjalizowanym oknie edycyjnym typu *outline* zawierającym hierarchicznie uporządkowane podstawowe fragmenty kodu.

Język SQLWindows zawiera podstawowe konstrukcje składniowe, takie jak instrukcje warunkowe, instrukcje wyboru, pętle, wywołania funkcji itp. Wyposażony jest również w mechanizmy specyficzne dla okienkowego środowiska zdarzeniowego: obsługa komunikatów, standardowe elementy interfejsu, typy i zmienne okienkowe itp.

Kategorie i typy klas

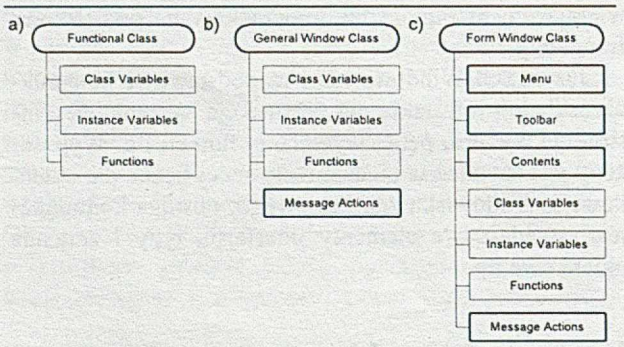
SQLWindows dzieli klasy na dwie główne kategorie: klasy funkcjonalne (ang. *functional classes*) i klasy okienkowe (ang. *window classes*).

Klasy okienkowe są oparte na standardowych elementach interfejsu. Za ich pomocą można zaimplementować dowolne okno główne (ang. *top-level window - Form Window, Table Window, Dialog Box, Quest Table Window*) i potomne (ang. *child window - Child Table Window, Push Button, List Box* itp.). W obrębie klas okienkowych wyróżnić można także klasy związane z innymi elementami graficznymi (*Line, Frame* itp.). W odniesieniu do klas okienkowych można posługiwać się pojęciem typu klasy. Typ klasy związany jest z tym, jaki element interfejsu dana klasa reprezentuje.

W przeciwieństwie do klas okienkowych, klasy funkcjonalne nie reprezentują elementów interfejsu. Są jedynie strukturami zawierającymi pola i metody. Opisany podział klas jest egzekwowany na poziomie składni SQLWindows.

Budowa klas w SQLWindows

Jak już wspomniano klasy funkcjonalne stanowią zbiór pól (ang. *Class variables, Instance variables*) i metod (ang. *Functions*). Klasy okienkowe dodatkowo zawierają listę atrybutów (ang. *Properties*) i sekcję obsługi komunikatów (ang. *Message actions*). W przypadku okien typu dialog (ang. *Dialog Box*), formatka (ang. *Form Window*) i tabela (ang. *Table Window, Quest Table Window*) jako składowe pojawiają się: pasek narzędzi (ang. *Toolbar*) oraz lista składników (ang. *Contents*). Składnikiem okna może być okno potomne, a także inny element graficzny. Formatka i tabela mają także składową opisującą menu okna (ang. *Menu*). Najważniejszą innowacją SQLWindows w stosunku do znanych języków obiektowych jest fakt pojawienia się w strukturze klasy nowych komponentów (na Rys. 1 wyróżniono je pogrubioną linią). Dotychczas klasa zawierała jedynie pola (zmienne) i metody (funkcje, procedury). W klasach okienkowych SQLWindows znalazły się dodatkowo składowe związane z właściwościami zdarzeniowego, graficznego systemu operacyjnego. Na uwagę zasługuje fakt, że wspomniane nowe składowe podlegają także dziedziczeniu, co zostanie szczegółowiej opisane w dalszej części artykułu.



Rys. 1. Składowe klas w SQLWindows a) klasa funkcjonalna, b) ogólna klasa okienkowa, c) klasa okienkowa

Pojawienie się takich wyspecjalizowanych komponentów pociąga jednak za sobą silne uzależnienie składni języka od specyfiki systemu operacyjnego, jakim jest w tym przypadku MS Windows.

Definiowanie klas i deklarowanie obiektów

Definicje klas są zawarte w sekcji *Application/Global Declarations/Class Definitions*. W tym miejscu znajdują się zarówno definicje klas funkcjonalnych, jak i okienkowych. Tworzenie definicji klas typu *Form Window, Dialog Box* i *Table Window* odbywa się wizualnie, szczególnie na etapie projektowania wyglądu okna i rozmieszczania na nim okien sterujących.

Deklaracja obiektu w SQLWindows wiąże się z utworzeniem obiektu danej klasy lub też z użyciem tegoż obiektu jako parametru funkcji lub okna. Sztywna struktura aplikacji występująca w tym języku pozwala na umieszczanie deklaracji obiektów jedynie w ściśle ustalonych miejscach kodu aplikacji. Reguły deklarowania obiektów są różne dla klas funkcjonalnych, klas okien głównych, klas okien potomnych i innych elementów graficznych.

Obiekty klas okien potomnych i innych elementów graficznych mogą stanowić zarówno składowe klas, jak i obiektów okien głównych.

Składowe klasowe i indywidualne

W SQLWindows rozróżnienie składowych klasowych i indywidualnych zrealizowano jedynie w odniesieniu do pól, co stanowi pewne ograniczenie, bowiem większość języków zorientowanych obiektowo pozwala definiować również metody klasowe i indywidualne. W SQL-Windows występują więc jedynie zmienne klasowe (ang. *Class variables*) i indywidualne (ang. *Instance variables*), zwane też instancyjnymi. Dopuszczalne jest odwoływanie się do zmiennych klasowych nawet wtedy, gdy nie został utworzony żaden obiekt danej klasy.

Edycja obiektów klas okienkowych

Kolejną cechą SQLWindows, nie spotykaną w innych zorientowanych obiektowo językach programowania, jest możliwość edycji obiektu. Cecha ta dotyczy jedynie kategorii klas okienkowych, lecz obejmuje wszystkie składowe obiektu. Oznacza to możliwość przeniesienia ostatecznej specjalizacji z ostatniego poziomu hierarchii klas na poziom obiektu klasy. Jest to możliwe tym bardziej, że poziom obiektów jest także objęty polimorfizmem. Przykładem wykorzystania mechanizmu edycji obiektów może być dostosowanie do bieżących potrzeb wyglądu zewnętrznego i struktury okna używanej klasy, bez konieczności ingerowania w definicję tej klasy lub definiowania dodatkowej klasy pochodnej. Możliwe jest także dodanie nowych funkcji do obiektu - funkcje te są

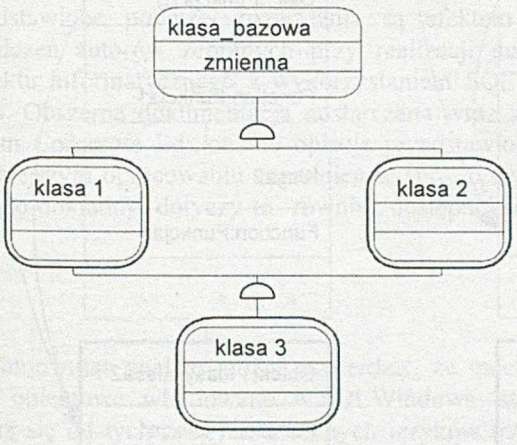
włączane w mechanizm polimorfizmu, co zostanie opisane później.

Dziedziczenie

Dziedziczenie w SQLWindows deklaruje się w sekcji *derived from*. W sekcji tej podajemy listę klas bazowych definiowanej klasy. Klasą bazową klasy funkcyjnej może być jedynie klasa funkcyjna. Klasą bazową klasy okienkowej mogą być:

- klasa funkcjonalna - co daje możliwość realizowania klas okienek ściśle powiązanych z klasami funkcyjnymi np. tworzenie okienek wyświetlających stan obiektu klasy funkcjonalnej,
- ogólna klasa okienkowa (ang. *general window class*) lub klasa okienkowa definiująca element tego samego typu co klasa bazowa. Takie ograniczenie dziedziczenia jest silnie uzasadnione - byłoby bowiem bezcelowe dziedziczenie np. klasy typu klawisz po klasie typu tabela.

Wspomniana wyżej ogólna klasa okienkowa została wprowadzona w SQLWindows od wersji 5.0. Jest ona swego rodzaju klasą abstrakcyjną - klasą okien, które nie mogą być wyświetlane na ekranie. Wprowadzenie klas ogólnych jest próbą udostępnienia takich mechanizmów, które umożliwiłyby łączenie odrębnych (pod względem wyglądu i działania) elementów interfejsu na wspólnym poziomie klasy bazowej. Wprowadzenie wspólnej, abstrakcyjnej klasy bazowej w tym przypadku pozwala na uniknięcie podwójnego programowania podobnych metod w obu klasach. Innym zastosowaniem klas abstrakcyjnych może być wykorzystanie sekcji *Message Actions* w celu zdefiniowania wspólnego sposobu reakcji na pewne komunikaty dla różnych elementów interfejsu (np. komunikat USR_CLEAR może oznaczać wyczyszczenie każdego z elementów interfejsu).

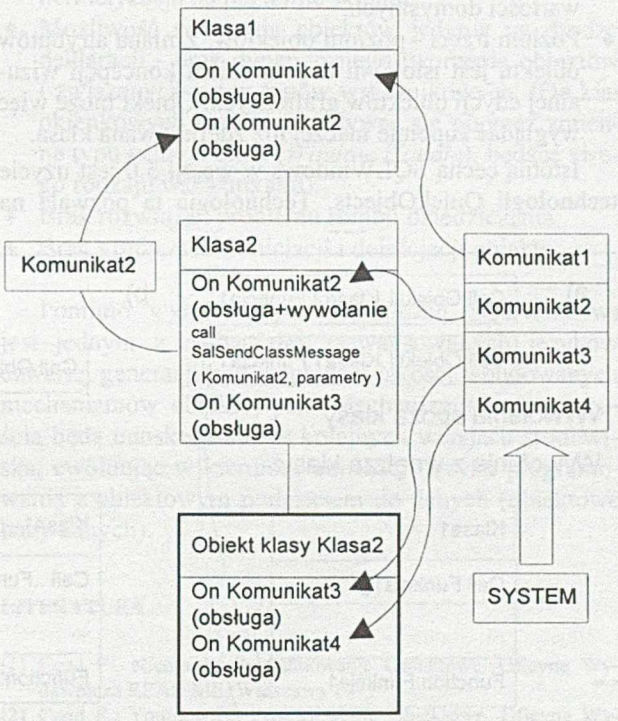


Rys.2. Ilustracja problemu rombu dziedziczenia

SQLWindows pozwala zarówno na dziedziczenie jednobazowe (ang. *single inheritance*) jak i wielobazowe (ang. *multiple inheritance*). Wadą dziedziczenia wielobazowego w SQLWindows jest jednak niemożność obejścia tzw. problemu rombu dziedziczenia (rys. 2). Klasa *klasa_bazowa* zawiera pole *zmienna*. Ponieważ z klasy tej dziedziczą klasy *klasa1* i *klasa2*, obie otrzymały pole *zmienna*. Klasa *klasa3* dziedziczy z klas *klasa1* i *klasa2*. Odwołanie do pola *zmienna* w obrębie klasy *klasa3* jest więc niejednoznaczne.

Dziedziczenie obsługi komunikatów

W klasach okienkowych SQLWindows występuje specyficzny mechanizm dziedziczenia, który określić można jako dziedziczenie obsługi komunikatów systemowych. Polega on na tym, że klasa pochodna dziedziczy reakcje na komunikaty zdefiniowane w klasach bazowych. W przypadku dziedziczenia wielobazowego sytuacja powielania się obsługi tego samego komunikatu w kilku klasach bazowych jest uznawana za błędną.



Rys.3. Obsługa komunikatów w hierarchii klas

Na uwagę zasługuje także fakt, że zdefiniowanie obsługi komunikatu na poziomie klasy pochodnej lub obiektu powoduje przesłonięcie tego samego komunikatu na poziomie klasy bazowej. Możliwe jest wywołanie obsługi tego komunikatu na wyższym poziomie poprzez przesłanie go jako tzw. komunikat klasowy (ang. *class message*) przy pomocy odpowiedniej funkcji (ang. *SalSendClassMessage*). Rys.3 ilustruje działanie mechanizmu dziedziczenia obsługi komunikatów na przykładzie struktury dwóch klas, *Klasa1* i *Klasa2* oraz obiektu *Obiekt1* klasy *Klasa2*. *Klasa2* jest klasą pochodną klasy

Klasa1. Wszystkie komunikaty przychodzące z systemu obsługiwane są na jak najniższym poziomie.

Dziedziczenie atrybutów

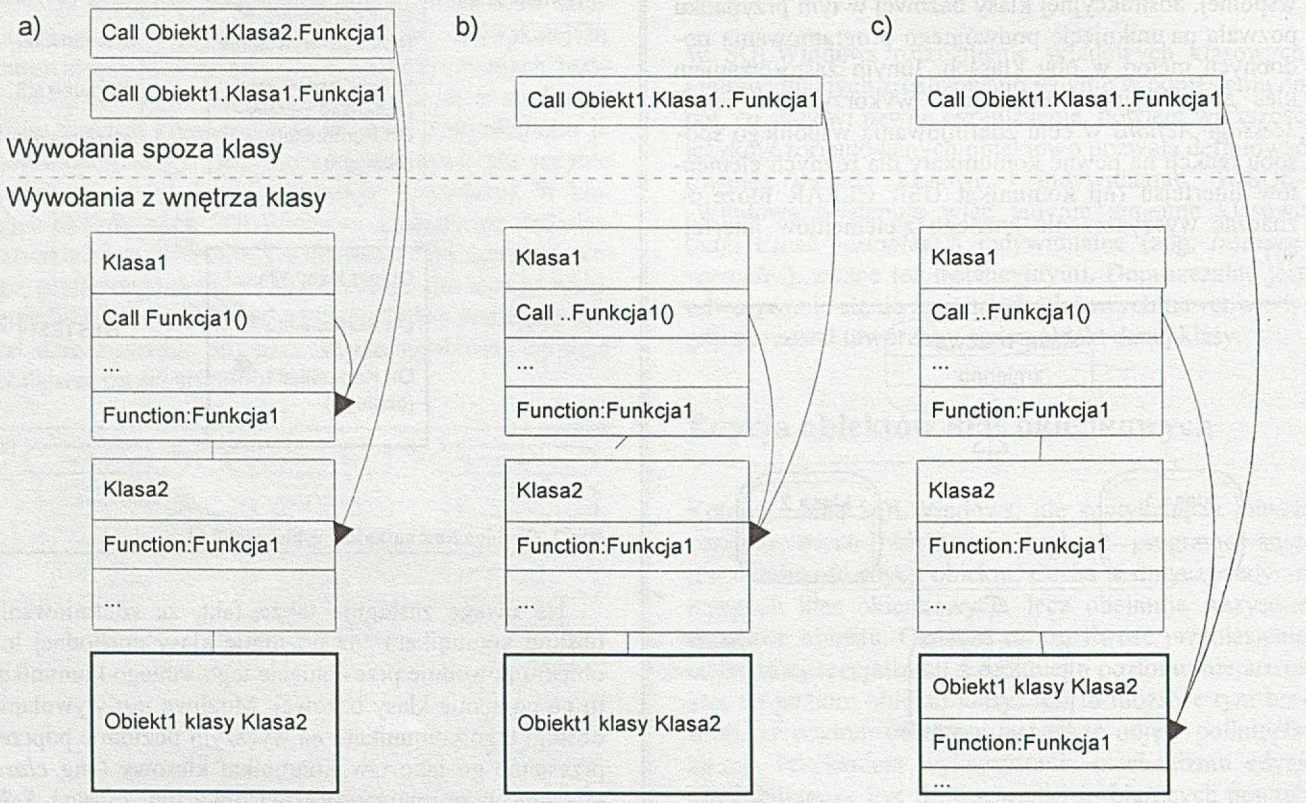
Klasy okienkowe i ich obiekty mają tzw. atrybuty (ang. *properties*). Atrybuty dotyczą zarówno wyglądu (czcionka, kolor, napis, ramka itp.), jak i zachowania się okna (maska wejściowa, automatyczna walidacja, sortowanie listy itp.). Mogą być one ustawiane zarówno podczas projektowania klas i edycji obiektów, jak również podczas działania aplikacji, przy użyciu odpowiednich funkcji. Atrybuty klas również podlegają dziedziczeniu. Mechanizm ten jest jednak nieco odmienny od mechanizmu dziedziczenia innych składowych klas. Przewiduje on następujące trzy stopnie:

- Stopień pierwszy - domyślny, ogólny wygląd okienek danego typu w całej aplikacji.
- Poziom drugi - klasy użytkownika. Na każdym z poziomów dziedziczenia jest możliwe modyfikowanie pewnych atrybutów albo pozostawienie im wartości domyślnych.
- Poziom trzeci - poziom obiektów. Zmiana atrybutów obiektu jest istotnym dopełnieniem koncepcji wizualnej edycji obiektów graficznych. Obiekt może więc wyglądać zupełnie inaczej niż zdefiniowana klasa.

Istotną cechą SQLWindows w wersji 5.0 jest użycie technologii QuickObjects. Technologia ta pozwala na dołączanie do listy atrybutów pewnych nowych, zdefiniowanych przez użytkownika parametrów, które w czasie działania programu mogą być wykorzystane do inicjacji obiektów (jako parametry metody inicjującej). Parametry te podlegają dziedziczeniu, podobnie jak atrybuty klas.

Hermetyzacja

SQLWindows nie zapewnia żadnych mechanizmów hermetyzacji na poziomie składni. Oznacza to, że komponenty klasy są dostępne dla wszystkich innych klas i funkcji bez żadnych ograniczeń. Doświadczenia autorów związane z tworzeniem bibliotek obiektowych dla SQLWindows pokazują jednak, że ograniczenie to nie stanowi istotnej wady języka, pod warunkiem przestrzegania zasad enkapsulacji na etapie projektowania i implementacji klas. Oczywiście brak kontroli kompilatora w zakresie udostępniania składowych klasy na zewnątrz jest kłopotliwy, ponieważ wiąże się z koniecznością samodzielnego jej prowadzenia. Najprostsza sugerowana metoda postępowania mająca na celu ułatwienie stosowania zasad hermetyzacji polega na sztucznym podzieleniu zmiennych klasy na dwa poziomy dostępu (prywatny, publiczny) i stosowaniu zasady, że do zmiennych prywatnych można się odwoływać jedynie za pośrednictwem akcesorów.



Rys.4. Niepolimorficzne i polimorficzne wywołania funkcji
a) odwołania niepolimorficzne, b) odwołania polimorficzne - funkcja zdefiniowana w klasie pochodnej, c) odwołania polimorficzne - funkcja zdefiniowana w obiekcie

Polimorfizm

Język SQLWindows jest wyposażony w specyficzny mechanizm odwołań polimorficznych. Każda funkcja zdefiniowana w klasie może zostać wywołana na dwa sposoby: niepolimorficznie (ang. *early-bound call*) i polimorficznie (ang. *late-bound call*). Aby wywołać funkcję polimorficznie, należy przed jej identyfikatorem umieścić znak {..} (dwie kropki) np. *Call..Funkcja()*, przy czym dotyczy to również odwołań kwalifikowanych np. *Call clsKlasa..Funkcja()*. Należy zwrócić uwagę na fakt, że taki sposób wywoływania funkcji sprawia, iż nie zachodzi konieczność określania a priori, czy funkcja będzie wywoływana polimorficznie, czy nie (co ma miejsce np. w C++ - funkcje wirtualne). W SQLWindows nie istnieją więc funkcje polimorficzne, ale polimorficzne wywołania funkcji. W przypadku wywołania polimorficznego sterowanie zostanie przekazane do funkcji o podanym identyfikatorze, znajdującej się najniżej w hierarchii klas, a w przypadku klas okienkowych nawet do funkcji zdefiniowanej w obiekcie (obiekty klas okienkowych objęte są polimorfizmem). Rys.4 ilustruje mechanizm wywołań polimorficznych na przykładzie struktury dwóch klas: *Klasa1* i *Klasa2* oraz obiektu *Obiekt1* klasy *Klasa2*. *Klasa2* jest klasą pochodną klasy *Klasa1*. W przypadku wywołania niepolimorficznego następuje przekazanie sterowania do funkcji zdefiniowanej w klasie, w której wywołanie wystąpiło lub do funkcji zdefiniowanej w klasie, której nazwy użyto jako kwalifikatora (Rys.4a). Drugi przypadek dotyczy wywołania polimorficznego dla tej samej struktury - sterowanie zostaje przekazane do funkcji zdefiniowanej w klasie znajdującej się najniżej w hierarchii (Rys.4b). Sterowanie zostanie przekazane do obiektu, jeżeli zostanie w nim zdefiniowana funkcja wywoływana polimorficznie (Rys.4c).

Podsumowanie

Przedstawione powyżej rozważania są efektem doświadczeń autorów zebranych przy realizacji dużego projektu informatycznego z wykorzystaniem SQLWindows. Obszerna dokumentacja dostarczana wraz z pakietem Corporate Edition nie opisuje przedstawionych w niniejszym opracowaniu zagadnień w sposób wystarczająco dokładny, dotyczy to również dostępnej literatury.

* * *

Podsumowując analizę można stwierdzić, że mechanizmy obiektowe wbudowane w SQLWindows istotnie różnią się od tych znanych z różnych języków trzeciej generacji. Podstawowe różnice dotyczą:

- Podziału klas na odrębne kategorie; podział ten zdaniem autorów stanowi jedną z pierwszych prób odzwierciedlenia różnic funkcjonalnych pomiędzy klasami na poziomie składni języka. Wynika to z faktu utrwalenia się podziału klas na grupy realizujące podstawowe składowe zadania - w przypadku SQLWindows taką grupę stanowią klasy związane z realizacją składowej kontaktu z użytkownikiem. W przyszłości jest możliwe pojawienie się na poziomie składni kolejnych wyspecjalizowanych klas służących - na przykład - do kontaktu z bazą danych.
- Pojawienia się nowych składowych w klasach, co jest efektem procesu podziału klas na specjalizowane kategorie.
- Możliwości edycji obiektu; cecha ta oznacza możliwość przeniesienia ostatecznej specjalizacji z najniższego poziomu hierarchii klas na poziom obiektu klasy.
- Objęcia poziomu obiektu mechanizmem polimorfizmu; pozwala to na przeniesienie na ten poziom najbardziej specjalizowanych usług.
- Rozbudowy mechanizmu dziedziczenia, polegającej na objęciu nim nowych składowych klas.

Mechanizmy obiektowe SQLWindows wykazują pewne ograniczenia. Do najważniejszych należą:

- Brak mechanizmów ochrony składowych klasy - hermetyzacja na poziomie składni.
- Możliwość tworzenia obiektów jedynie na drodze deklaracji - brak dynamicznego tworzenia obiektów i związanych z tym typów wskaźnikowych. (Do klas okienkowych można odwoływać się poprzez zmienne typu uchwyt okna (*Window Handle*), będące swego rodzaju wskaźnikami).
- Brak rozwiązań problemu rombu dziedziczenia.
- Brak grupy metod inicjacji i deinicjacji obiektu.

Pomimo wymienionych ograniczeń SQLWindows jest jednym z najbardziej zaawansowanych języków czwartej generacji pod względem jakości wbudowanych mechanizmów obiektowych. Mechanizmy te z pewnością będą udoskonalane w kolejnych wersjach środowiska, ewoluując w kierunku unifikacji języka programowania z obiektowym podejściem do danych (obiektowe bazy danych).

LITERATURA

[1] Coad P., Nicola J.: Programowanie Obiektowe. Oficyna Wydawnicza READ ME, Warszawa 1993
[2] Coad P., Yourdon E.: Projektowanie Obiektowe. Oficyna Wydawnicza READ ME, Warszawa 1994
[3] Gietz W.: SQLWindows Programming, Gupta Corporation 1993,
[4] Holmes-Kinsella D.: Using Gupta SQLWindows 5. QUE Corporation 1995
[5] Riny B.: SQLWindows version 5.0 Developers Reference. Gupta Corporation 1994.

Jacek SZEDEL i **Michał KOLANO** są stypendystami doktoranckimi w Zakładzie Opraogramowania Instytutu Informatyki Politechniki Śląskiej w Gliwicach.
Czytelników zainteresowanych tematyką przedstawioną w artykule prosimy o kontakt.
E-mail:
jszedel@star.iinf.polsl.gliwice.pl,
mkolano@star.iinf.polsl.gliwice.pl.

Koncepcja modelu dla obiektowej bazy danych

Maciej Piasecki
Centrum Informatyczne Politechnika Wrocławska

Pojęcie **obiekt** jest powszechnie stosowane, różnorodnie interpretowane i definiowane. Utrudnia to porozumienie w tych dziedzinach informatyki, które posługują się tym pojęciem. Szczególnie dotyczy to **obiektowych baz danych**, dlatego też chciałbym przedstawić w tym artykule definicje kilku podstawowych pojęć, jak również opierający się na nich model logiczny dla obiektowego systemu zarządzania bazą danych OSZBD.

Czym jest OSZBD?

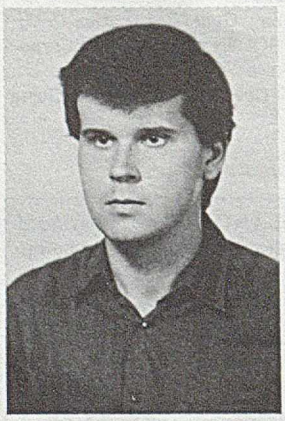
Zastosowania obliczeniowe we współczesnej informatyce stanowią grupę marginalną. Dominuje przetwarzanie informacji, jej przechowywanie i udostępnianie. Znaczną część rynku oprogramowania stanowią SZBD (Systemy Zarządzania Bazą Danych) oraz bazujące na nich aplikacje. W obrębie tych pierwszych najczęściej spotykanym terminem jest pojęcie *relacyjny*.

Jest to skojarzenie jak najbardziej słuszne – na technologii relacyjnej są oparte wszystkie obecnie najbardziej znane i popularne komercyjne SZBD. Warto sobie jednak przypomnieć, że model relacyjny został zaproponowany przez Codd'a w 1970 roku [1]. Nie była to propozycja czysto formalna, ale konkretne, praktyczne rozwiązanie bazujące na doświadczeniach w stosowaniu modelu hierarchicznego i sieciowego. Model relacyjny szybko zdobył sobie popularność. Lata osiemdziesiąte to jego całkowita dominacja. Praktyczne zalety modelu relacyjnego spowodowały jego sukces, ale też stały się przyczyną wielu jego „wad wrodzonych" (obszerna analiza tego zagadnienia znajduje się w pracy [5]).

Wystarczy przypomnieć sobie, jak wyglądała informatyka stosowana dwadzieścia pięć lat temu. Minęła cała epoka! Model relacyjny jest odbiciem ówczesnych potrzeb i wymagań. Moc przetwarzania i pojemności pamięci były stosunkowo niewielkie. Ograniczało to możliwości przetwarzania informacji. Aplikacje zarządzały dość prostymi danymi. Używane języki programowania nie były wyposażone w możliwości konstruowania skomplikowanych abstrakcyjnych typów danych. Komunikacja z użytkownikiem stanowiła dla systemów zadanie marginalne. Obecnie sytuacja uległa zasadniczej zmianie. Dzisiejsze wymagania i oczekiwania użytkowników uległy zwiększeniu. Wzrósł także znacznie poziom komplikacji systemów. Obsługa interfejsu użytkownika pochłania do 60% czasu procesora. Jakość interfejsu użytkownika ma ciągle rosnący wpływ na ocenę oprogramowania. Pojawiła się konieczność przetwarzania danych multimedialnych (tekst, grafika, dźwięk, animacja, itd.).

Warto przypomnieć kilka najistotniejszych ograniczeń stosowalności modelu relacyjnego w zmienionej sytuacji.

- Wielką zaletą modelu relacyjnego jest jego prostota. Dużo na tym zyskuje łatwość i efektywność implementacji. Niestety jednocześnie stało się to jego poważną wadą: model relacyjny nie przystaje do modeli danych używanych przez współczesne języki programowania.
- Model relacyjny nie jest w stanie w naturalny sposób zaakceptować rozbudowanych struktur danych, co ujawnia się już w konfrontacji z językami programowania wcześniejszej generacji niż obiektowe. Pierwsza postać normalna zakłada atomowość atrybutów.
- Brak w nim możliwości definiowania przez użytkownika własnych abstrakcyjnych typów danych przechowywanych w bazie danych. W połączeniu z odmiennością języka manipulacji danymi i języka programowania, użytego w aplikacji, powstaje przepaść pomiędzy aplikacją a SZBD. Zalety nowoczesnego języka programowania giną przy przekroczeniu granicy bazy danych.
- Do powszechnego użycia weszły różnorodne metody analizy i projektowania obiektowego [6]. Wyprodukowany przy ich zastosowaniu projekt wymaga jednak konwersji do wymagań modelu relacyjnego. Najpoważniejszym problemem staje się rozróżnialność obiektów użytych do modelowania świata. W większości metodologii dwa obiekty są rozróżnialne bez względu na ich stan wewnętrzny. Niestety krotka w relacji istnieje jedynie poprzez wartości swoich atrybutów. Aby temu zaradzić w modelu relacyjnym już od dawna wykorzystuje się pojęcie **klucza unikalnego** – ale ten jest niestety zależny od wartości atrybutów.



Mgr inż. Maciej PIASECKI ukończył w 1993 roku Wydział Elektroniki Politechniki Wrocławskiej, o kierunku Informatyka. Od 1994 roku jest pracownikiem Centrum Informatycznego PWr. Prowadzi badania w zakresie: przetwarzania języka naturalnego, automatycznego tłumaczenia tekstów, obiektowych baz danych i wykorzystania technik multimedialnych. Zaprojektował, a następnie kierował zespołem realizującym obiektowy system zarządzania bazą danych *OB! OK!*, który znalazł zastosowanie praktyczne w multimedialnym systemie informacyjnym SMOK, tworzonym dla PKP. Laureat II nagrody w 1993 r. w organizowanym corocznie przez PTI ogólnopolskim konkursie na najlepszą pracę magisterską z dziedziny informatyki.

- Brak możliwości obsługi danych multimedialnych w RSZBD (Relacyjny System Zarządzania Bazą Danych).

Lista wad i ograniczeń mogłaby być dłuższa, ale już te wymienione ukazują najważniejsze trudności wynikające ze stosowania modelu relacyjnego. Zapewne to one stały się powodem badań nad rozszerzeniem modelu relacyjnego lub nawet zastąpienie go innym modelem, prezentującym zupełnie nowe podejście koncepcyjne.

Obiektowy system zarządzania bazą danych

Pierwszym krokiem w kierunku nowych rozwiązań stały się próby skonstruowania modelu poszerzającego model relacyjny w kierunku złamania warunku pierwszej postaci normalnej, czyli dające możliwość definiowania atrybutów o typie strukturalnym [5].

Kłopoty związane ze stosowaniem mechanizmu **klucza unikalnego** próbowano rozwiązać przez wprowadzenie, dla elementów przechowywanych w bazie, unikalnego identyfikatora (język *LDM* [7]), niezależnego od wartości atrybutów oznaczanej krotki.

W systemie *Postgres* [10] natomiast podjęto próbę wprowadzenia do RSZBD możliwości definiowania własnych abstrakcyjnych typów danych. Rozwiązanie to opierało się na idei podwójnej reprezentacji danych:

- wewnętrznej, określającej format, w którym dane są przechowywane w bazie,
- zewnętrznej, widocznej dla użytkownika.

Pełna definicja składała się ze specyfikacji obydwu postaci, odpowiednich funkcji konwersji oraz zdefiniowanych dla postaci zewnętrznej operacji, które były możliwe do wykonania na danym typie (w jego postaci zewnętrznej). Definicje te były dołączane do RSZBD w postaci modułu napisanego w zewnętrznym (w stosunku do systemu) języku programowania.

W ten sposób przygotowano elementy niezbędne do dokonania dalszego postępu. Ostatecznego impulsu dostarczył rozwój obiektowych języków programowania. W dziedzinie wytwarzania oprogramowania języki obiektowe przyniosły ogromny postęp jakościowy. W konsekwencji, w 1984 roku został uruchomiony pierwszy prototyp obiektowego systemu zarządzania bazą danych. Był to system *Gemstone* [4], będący rozszerzeniem pierwszego obiektowego języka programowania *Smalltalk*. Specyficzne cechy wzorca stanowiły jednak największą słabość całego systemu [5].

Czym różni się OSZBD od tradycyjnego RSZBD?

Pełną charakterystykę OSZBD można znaleźć w pracy [13]. Przypomnijmy tutaj jego najistotniejsze cechy.

Pomimo dużej różnorodności w zastosowanych modelach logicznych i rozwiązaniach konstrukcyjnych w istniejących OSZBD, można wskazać kilka podstawowych cech, które spotyka się najczęściej. Większość systemów obiektowych radzi sobie z ograniczeniami RSZBD. Informacja jest przechowywana w postaci **obiektów** (odpowiadających **bytom** z obiektowych metodologii stosowanych w projektowaniu) pogrupowanych w **klasy**. O przynależności obiektu do klasy decyduje zarówno jego struktura wewnętrzna, jak i posiadane przez niego **metody**, tzn. zachowanie (sposób obsługi). Dzięki temu klasa wypełnia zadania **abstrakcyjnego typu danych**. Z założenia większość OSZBD daje możliwość zarządzania danymi multimedialnymi.

Największą wadą obiektowych systemów zarządzania jest mała efektywność ich działania (w szczególności efektywność implementacji metod obiektów) oraz ciągle niedostateczny rozwój teorii leżącej u podstaw OSZBD. Dotyczy to zarówno modelu logicznego, jak i aspektów technicznych (np. optymalizacja dostępu do bazy) [5].

Niewątpliwie niedomagania OSZBD są spowodowane wczesną fazą ich rozwoju. W przeciwieństwie do stabilizacji modelu relacyjnego w zakresie rozwiązań stosowanych, w modelu obiektowym panuje ciągle duża różnorodność i konkurencja. Nie ma wypracowanego jednolitego standardu. Spotykane rozwiązania reprezentują zarówno wpływ obiektowych języków programowania, jak i obiektowych metod analizy i projektowania.

Różnorodne SZBD pretendujące do miana *obiektowych*, pod względem realizacji *obiektowości* można podzielić na trzy grupy (rys.).

Klasyfikacja obiektowych systemów zarządzania bazą danych

- Pierwsza grupa to systemy, gdzie w bazie danych są przechowywane **elementy** o dowolnej strukturze wewnętrznej, mogące wchodzić we wzajemne związki, jednoznacznie identyfikowane przez **unikalny identyfikator**, ale nie posiadające **metod** (dlatego też używam słowa element, a nie obiekt).
- Grupa druga właściwie ma jednego reprezentanta. Jest nim system *Illustra* [9] będący bezpośrednim rozwinięciem *Postgres'u* [7]. Definiowane w nim własne **abstrakcyjne typy danych** przybrały postać **klasy**. Jednak mechanizm leżący u podstaw został ten sam – podwójna reprezentacja. Została ponadto dodana możliwość przechowywania danych o znacznych rozmiarach (np. multimedialnych). Główna idea reprezentuje bardzo praktyczne podejście – chęć zachowania sprawdzonych mechanizmów starszej wersji relacyjnej. System jest określany przez autorów jako realizacja kroku pośredniego pomiędzy modelem relacyjnym a obiektowym.
- Ostania grupa to OSZBD w pełnej postaci, z możliwością definiowania **metod obiektów** łącznie.

Wybór modelu dla OSZBD

Brak jednolitej terminologii w zakresie *obiektowości* systemów prowadzi czasami do nieporozumień, różnorodność stosowanych rozwiązań i ich konkurencyjność

są jak najbardziej pozytywne. Niestety niedomagania OSZBD pod względem efektywności stanowią ich bardzo poważną wadę. Praktycznie uniemożliwia to konkurencję z systemami relacyjnymi. Możliwe, że pewnym praktycznym rozwiązaniem są systemy sklasyfikowane tutaj jako **relacyjno-obiektowe**. Wydaje się jednak, że rozwiązanie to omija istotę problemu.

OSZBD są nie tylko wolne. Wymagają do swojej pracy pokaźnych zasobów sprzętowych: mocy przetwarzania i pamięci operacyjnej. Paradoksalnie jest to w znacznej mierze spowodowane bogactwem realizowanego modelu obiektowego. Model obiektowy jest najczęściej oparty na wzorcach wypracowanych w metodologiach analizy i projektowania obiektowego. Wydaje się, że jest to „wada wrodzona" OSZBD. Czy jednak tak być musi?

Jedną z cech negatywnych modelu relacyjnego był brak integracji języka manipulacji danymi z językiem programowania aplikacji. Większość OSZBD proponuje rozwiązanie tego problemu w drodze zastosowania jednego, wspólnego **uniwersalnego języka programowania**. Podstawą jest tam **trwałość** tworzonych obiektów, ich zdolność do istnienia pomiędzy kolejnymi sesjami.

W większości OSZBD uniwersalny język programowania jest skonstruowany w sposób „systemocentryczny". Oznacza to tworzenie nowych wersji istniejących już języków programowania np. *C* (*CO_2* [5], *Ontos* [2]). W gruncie rzeczy prowadzi to do powstania zupełnie nowego tworu [5], wraz ze wszystkimi niekorzystnymi konsekwencjami (rynek gotowych modułów kodu, bibliotek, specyficzne kompilatory).

Rozważając zarysowane powyżej sprzeczności:
- efektywność – bogactwo modelu,
- integracja – zachowanie standardu,

można dojść do wniosku, że dobrym wyjściem z zaistniałej sytuacji byłoby zbudowanie OSZBD opartych na znacznie prostszym modelu logicznym. Model ten byłby przez to łatwiejszy do implementacji i w zasadzie wzorowany na modelu obiektowym zaczerpniętym z powszechnie stosowanych imperatywnych obiektowych języków programowania. W ten sposób językiem OSZBD mógłby być jeden z ogólnie stosowanych obiektowych języków programowania. Próba takiego podejścia zostanie przedstawiona w dalszej części artykułu.

Opis formalny modelu obiektowego

Dotychczas nie ma jednolitej, powszechnie akceptowanej, formalnej definicji **obiektu, klasy** oraz pojęć od nich pochodnych. Są one definiowane w zależności od potrzeb. Różnice w definicjach występują nawet w obrębie jednej dziedziny, takiej jak bazy danych, mimo powszechnego intuicyjnego zrozumienia. Dlatego dla opisu podstawowych cech proponowanego modelu obiektowego OSZBD spróbujmy uściślić definicje pojęć używanych w jego opisie.

Byt (ang. *entity*) – dowolny rzeczywisty lub abstrakcyjny element świata lub – zawężając – modelowanego wycinka rzeczywistości.

Obiekt – model **bytu**. Jest on charakteryzowany przez swoje zachowanie (**metody**) i swój **stan**. Każdy **obiekt** można odróżnić od dowolnego z pozostałych. **Obiekt** posiada własność **hermetyzacji**.

Hermetyzacja (lub też enkapsulacja, ang. *encapsulation*) – każda zmiana **stanu** obiektu może zajść jedynie w wyniku działania dowolnej z **metod obiektu**.

Akcja – coś co się wydarza, zbiór akcji powiązanych z obiektem dzieli się na:

akcje wewnętrzne – zachodzą one bez udziału środowiska (pozostałych elementów modelu), oraz

interakcje – wykonywane z udziałem środowiska, ich wykonanie jest inicjowane przez środowisko.

Metoda – sposób obsługi przez obiekt akcji wewnętrznej (**metoda prywatna**) lub interakcji (**metoda publiczna**).

Typ obiektowy – predykat charakteryzujący zbiór obiektów. Dowolny obiekt x jest typu A wtedy i tylko wtedy, gdy spełnia predykat **typu obiektowego** A.

Klasa – zbiór wszystkich **obiektów** spełniających pewien **typ obiektowy**. Każda **klasa** jest wyznaczana przez dokładnie jeden **typ obiektowy**.

Podtyp – typ obiektowy A jest **podtypem** dla typu B i B jest **nadtypem** dla A jeżeli każde x, takie że spełnia **typ obiektowy** A, spełnia również **typ obiektowy** B.

Podklasa, nadklasa – jedna **klasa** jest **podklasą** drugiej i odwrotnie, wtedy i tylko wtedy, gdy **typ obiektowy** skojarzony z pierwszą z nich jest **podtypem typu obiektowego** skojarzonego z drugą. Kiedy jedna klasa jest **podklasą** drugiej, to mówimy o **dziedziczeniu klas**.

Kreacja (lub też konstrukcja) – realne zaistnienie w modelu wycinka rzeczywistości **obiektu** określonej **klasy**.

Destrukcja – usunięcie **obiektu** z modelu wycinka rzeczywistości.

Polimorfizm – własność polegająca na tym, że dana **klasa** jest **podklasą** kilku innych.

Skomentowania wymaga użyte w definicji **typu obiektowego** określenie *obiektowy*. **Typ obiektowy** nie stanowi bowiem równoważnika dla **abstrakcyjnego typu danych**. Jeżeli jednak zawęzimy to pojęcie, wymagając od predykatu określającego typ definiowania zarówno własności obiektów (co wyznacza nam pewien ich zbiór), jak i operacji możliwych do wykonania na tym zbiorze, to ta „uściślona" wersja definicji jest już w pełni zgodna z definicją **abstrakcyjnego typu danych**. Formalny dowód tego stwierdzenia wybiega poza ramy tego artykułu.

Korzystając ze zdefiniowanych pojęć podstawowych można określić pełną charakterystykę modelu. Kluczowym elementem **modelu obiektowego** jest **obiekt** – najmniejsza rozróżnialna część konstruowanego obrazu wy-

cinka rzeczywistości. Jest on, zgodnie z definicją, charakteryzowany przez jeden z możliwych **typów obiektowych** spełniających poniższy schemat.

Typ obiektowy wyznacza zbiór elementów – **klasę obiektów** $O:\{(s,i): s \in S, i \in I\}$ gdzie:

stan (obiektu) – $s \in S \equiv \{<a_1:v_1 \in T_1, a_2:v_2 \in T_2, ..., a_n:v_n \in T_n>\}$, gdzie $n \in N$ (zbioru liczb naturalnych), a_i jest nazwą atrybutu obiektu, v_i jego wartością typu T_i gdzie, T_i może być jednym z **typów prostych** lub **złożonych** zdefiniowanych na potrzeby *C++* lub też dowolnym typem złożonym skonstruowanym w oparciu o prezentowaną tutaj definicję.

identyfikator (obiektu) – $i \in I$ gdzie I jest dowolnym nieskończonym, przeliczalnym zbiorem (dowolne dwa elementy są od siebie rozróżnialne). W **modelu obiektowym** jest określone wzajemnie jednoznaczne, różnowartościowe odwzorowanie „na" P przypisujące **obiektom** dowolnego **typu obiektowego** wartości ze zbioru I.

Dla zbioru **obiektów** określonego **typu obiektowego**, czyli w myśl przyjętej uprzednio definicji **klasy**, typ obiektowy określa również zbiór możliwych do wykonania funkcji, czyli

metod – stanowiących zbiór $F \equiv \{m_1:f_1, m_2:f_2, ..., m_k:f_k\}$, gdzie $k \in N$, m_i jest **selektorem** metody a f_i jest dowolnym odwzorowaniem spełniającym następujący schemat:

$f_i:\{s_0 \in S, x_1 \in T_1, ..., x_m \in T_m\} \rightarrow \{s_1 \in S, y_1 \in T_1, ..., y_t \in T_t\}$, gdzie s_0 jest stanem obiektu, do którego została zastosowana metoda, s1 jest nowym stanem obiektu zmienionym przez działanie metody (w szczególnym przypadku obydwa te stany mogą być sobie równoważne) oraz m, $t \in C$ (liczb całkowitych) i m, $t \geq 0$.

Warto tutaj podkreślić niewidoczną może na pierwszy rzut oka cechę metody: nie zmienia ona nigdy wartości identyfikatora obiektu.

Według zdefiniowanego powyżej **typu obiektowego** można zinterpretować pojęcie:

Dziedziczenie – obiekt należący do **podtypu** musi spełnić wymagania **nadtypu**, czyli jego struktura wewnętrzna i zbiór możliwych do wykonania operacji musi zawierać przynajmniej te elementy, które zostały już określone przez **predykat nadtypu**, z dokładnością co do nazw, typów **atrybutów** oraz **selektorów** i specyfikacji **metod**; **klasa** dziedzicząca zawiera się w **klasie**, po której dziedziczy.

Wprowadźmy definicje dwóch kolejnych pojęć modelu obiektowego:

Agregacja – składanie się jednego obiektu z innych obiektów. Z agregacją mamy do czynienia wtedy, gdy występuje przynajmniej jeden atrybut **typu obiektowego**.

Asocjacja – powiązania występujące pomiędzy obiektami, ale innej natury niż agregacja (wiedza o istnieniu drugiego obiektu), co może być wyrażone przez atrybut typu I (zbiór identyfikatorów) lub typu $I \times O$ (iloczyn kartezjański zbioru identyfikatorów z dowolną **klasą**). Pierwsze rozwiązanie można interpretować jako *wskazanie nieokreślone*, natomiast w wyniku drugiego otrzymujemy *wskazanie o określonym typie*.

Zatem, zgodnie z naszą intencją, udało się uzasadnić, że zdefiniowany powyżej model obiektowy ma wszelkie cechy podobieństwa do modelu będącego podstawą powszechnie używanych imperatywnych obiektowych języków programowania, czyli *C++*, *Pascal* i im podobnych. Równocześnie pojęcia **asocjacji** i **agregacji** zbliżają proponowany model obiektowy OSZBD do modeli leżących u podstaw obiektowych metodologii analizy i projektowania systemów.

Perspektywy zastosowania zdefiniowanego modelu

Co wnosi zastosowanie programistycznego modelu obiektowego jako podstawy działania OSZBD?

- Niewątpliwie zdefiniowany model obiektowy stanowi znaczne uproszczenie w stosunku do modeli wywodzących się z obiektowej analizy lub też obiektowych metodologii projektowania systemów. Jego prostota daje większe szanse na efektywną realizację OSZBD i zmniejszenie wymagań co do zasobów sprzętowych.
- Z natury „bliskości pojęciowej", model obiektowy dokonuje ścisłej integracji języka programowania aplikacji z językiem manipulacji danymi. Mamy tu do czynienia z przeniesieniem struktury obiektu z pamięci operacyjnej komputera do pamięci stałej. Obiekt o charakterze **trwałym** jest definicyjnie identyczny z obiektem „programistycznym".
- W trakcie projektowania aplikacji systemu informacyjnego przestaje być problemem rozgraniczenie pomiędzy *obiekty programu* a *obiekty danych*. Mamy do czynienia z jedną, spójną, abstrakcyjną przestrzenią systemu.
- Znikają omówione wcześniej ograniczenia i wady wynikające ze stosowania relacyjnego modelu SZBD.

Jakie problemy mogą się ujawnić przy wykorzystaniu tak zbudowanego OSZBD?

- Podstawową wadą omówionego modelu obiektowego jest to, że zastosowane w nim pojęcie obiektu nie przystaje w pełni do pojęcia obiektu w obiektowych metodologiach analizy i projektowania systemów. Metodologie te posługują się o wiele mocniejszym pojęciem obiektu. Przejście od projektu logicznego do projektu technicznego aplikacji wymaga więc konwersji (np. problemy z mechanizmem wersyjności obiektów). Jednak różnice pojęciowe nie są aż tak duże, jak w przypadku stosowania modelu relacyjnego.
- Inne mankamenty stanowią ograniczenia w operacjach na danych, które jednak można łatwo pokonać. Nic nie stoi na przeszkodzie, aby na poziomie OSZBD definiować relacje porządkujące, określone na obiektach danej klasy, oraz za pomocą tych relacji tworzyć uporządkowane kolekcje obiektów. Duże możliwości kryją się w efektywnym wykorzystaniu mechanizmów dziedziczenia, agregacji i asocjacji.

* * *

O przydatności kreowanych projektów zawsze decyduje praktyka. Dopiero po implementacji danego projektu można mówić o sukcesie lub porażce zamierzenia.

dokończenie na s. 32

Model edukacji informatycznej społeczeństwa

Marek Miłosz
Katedra Infromatyki
Politechnika Lubelska

Informatyka z jej narzędziami wkracza do każdej dziedziny naszego życia. To, co jeszcze parę lat temu było dostępne tylko dla wąskiej grupy osób, staje się powszechne. Coraz częściej od poziomu umiejętności obsługi komputera zależy otrzymanie lepiej płatnej pracy czy też sukces życiowy pracowników w różnych zawodach. Informatyka wkracza także do domów. Początkowo pod postacią gier, a ostatnio coraz częściej jako komputery domowe i Internet.

Jest rzeczą oczywistą, że model kształcenia informatycznego musi ulec w najbliższym czasie dużym przeobrażeniom. Nauczanie informatyki nie może pozostać domeną wyższych uczelni i niektórych elitarnych szkół średnich. Model kształcenia informatycznego powinien stawić czoła wyzwaniom niesionym przez czas i zapewnić wykształcenie na odpowiednim poziomie. Jak powinien wyglądać zatem ten model w Polsce? Problemowi temu poświęcone były V Jubileuszowe Warsztaty Mikrokomputerowe Politechniki Lubelskiej, które odbyły się w Kazimierzu Dolnym w dniach 22–26 kwietnia br.

Edukacja informatyczna rozdziela się na dwa rozłączne z pozoru kierunki: kształcenie zawodowych informatyków i nauczanie szeroko rozumianych użytkowników systemów informatycznych. Nie umniejszając wagi pierwszego kierunku, należy podkreślić, że użytkowników jest znacznie więcej. Do tej grupy należy bowiem ekspedientka w sklepie, obsługująca kasę fiskalną lub terminal systemu sprzedaży, sekretarka pisząca przy pomocy edytora tekstów i mikrokomputera PC, a także mechanik samochodowy testujący silnik specjalnym komputerowym urządzeniem, nie mówiąc już o inżynierze-projektancie czy też współczesnym menedżerze. Trudno zresztą wymienić wszystkie zawody, których przedstawiciele aktywnie wykorzystują systemy informatyczne. Problemy kształcenia informatycznego w tej grupie są znaczne. Z jednej strony są związane z masowością, a z drugiej z dużym zróżnicowaniem potrzeb.

Problemy edukacji informatycznej

W związku z powyższym wydaje się, że **model edukacji informatycznej społeczeństwa powinien wynikać z potrzeb kształcenia użytkowników, a nie informatyków zawodowych.**

Dwa kierunki edukacji informatycznej: kształcenie informatyków zawodowych i użytkowników mają także część wspólną. Jest nią szkolnictwo podstawowe i średnie. Ogólny poziom wiedzy informatycznej społeczeństwa zależy od przygotowania młodzieży szkolnej. Z tym związany jest także poziom nauczania i stopień jego realizacji w szkole wyższej.

W chwili obecnej kładzie się duży nacisk na edukację informatyczną na poziomie szkoły podstawowej i średniej. Znane problemy – głównie pieniądze i braki kadrowe, ale także brak sensownych programów metodycznych – doprowadziły jednak do dużego zróżnicowania w poziomie absolwentów szkół średnich. Absolwenci klas informatycznych są dość dobrze przygotowani, inni natomiast nawet nie zetknęli się w szkole z komputerem. Do życia zawodowego i na studia wyższe trafiają zatem osoby, które świetnie (i nie tylko manualnie) władają niektórymi narzędziami informatycznymi i takie, których wiadomości i umiejętności są praktycznie zerowe. Przy utrzymującym się tradycyjnym i mało elastycznym sposobie kształcenia na studiach wyższych powoduje to ogromne trudności w planowaniu i realizacji zajęć, nie tylko informatycznych. Coraz częściej bowiem student używa komputera w trakcie zajęć z przedmiotów nie związanych z informatyką. Jeśli nie potrafi go obsługiwać, jest pozbawiony kolejnego narzędzia do zdobywania wiedzy.

Szkolnictwo podstawowe i średnie

Podstawowymi brakami szkolnictwa podstawowego i średniego w dziedzinie kształcenia informatycznego są:

- niedostateczny poziom ilościowy i jakościowy wyposażenia szkół w sprzęt komputerowy i oprogramowanie,
- duże braki kadrowe,
- niespójne i niemożliwe do realizacji programy nauczania.

W chwili obecnej zaledwie 45% szkół średnich posiada jakikolwiek sprzęt komputerowy. Centralne zakupy realizowane przez MEN, częstokroć przy wsparciu różnorodnych programów zagranicznych, są kroplą w morzu potrzeb. Większość szkół ma sprzęt komputerowy dzięki ofiarności rodziców lub przedsiębiorczości nauczycieli i uczniów. Nawet jeśli szkoła posiada już pracownię komputerową, to ogromnym problemem jest organizacja udostępniania jej uczniom. Brakuje funduszy na nadgodziny dla nauczycieli i zatrudnienie specjalistów.

Zła sytuacja finansowa szkolnictwa nie sprzyja pozyskiwaniu dla niego nowych kadr. Absolwenci kierunków informatycznych, wobec ogromnego popytu na rynku

pracy, pracę w szkole podejmują w ostatniej kolejności. Istnieje jednak korzystne rozwiązanie kadrowe. Przez system podnoszenia lub zmiany kwalifikacji nauczyciele różnych przedmiotów (głównie matematyki, fizyki i techniki) są w stanie uzyskać uprawnienia do nauczania przedmiotu *Podstawy informatyki* w szkole średniej. Podyplomowe studia informatyczne dla nauczycieli prowadzone są w wielu ośrodkach w kraju (np. Wrocław, Poznań, Gdańsk, Kielce czy też Lublin [3]). Są to dość specyficzne studia podyplomowe – nie służą bowiem podnoszeniu kwalifikacji absolwentów kierunków informatycznych, lecz kształcą na informatyków absolwentów innych kierunków. W związku z tym mają one umożliwić nauczycielowi poznanie informatyki od podstaw. Stąd też duży nacisk kładzie się na tych studiach na wiedzę podstawową i metodologiczną [2]. Wiedza i umiejętności praktyczne w informatyce bardzo szybko się starzeją. Nauczyciel informatyki powinien stale ją uzupełniać na różnego rodzaju kursach i szkoleniach. System takich ośrodków szkoleniowych powstał w oświacie przy Wojewódzkich Ośrodkach Metodycznych.

Wobec dużego zapotrzebowania na nauczycieli podstaw informatyki w szkołach średnich (nie wspominając już o podstawowych), ocenianych na co najmniej 4000 nauczycieli w okresie 1996–1997 [6], studia podyplomowe nie są w stanie, moim zdaniem, wykształcić wystarczającej liczby nauczycieli. Oświatę polską czeka więc w najbliższym czasie jeszcze większy niedobór kadr w zakresie nauczania podstaw informatyki. Sytuacja wśród nauczycieli innych przedmiotów, którzy powinni wykorzystywać komputer w procesie dydaktycznym, jest jeszcze gorsza. Nie będzie tych niedoborów tylko w przypadku, gdy założenia programu [6], związane z zakupami sprzętu, nie będą zrealizowane.

Zmieniająca się technologia informatyczna wymaga wnoszenia coraz to nowych treści do nauczania informatyki w szkołach średnich i rozpoczęcia tego procesu już na poziomie szkół podstawowych. To z kolei wymaga opracowania i realizacji spójnego programu w dziedzinach wyposażenia technicznego, kadrowego i metodycznego szkolnictwa. Taki ramowy program powstał [6] w Ministerstwie Edukacji Narodowej, jednak nie wiadomo, czy wystarczy funduszy na jego realizację.

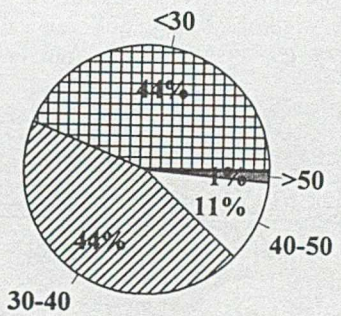
Szkolnictwo wyższe i studia podyplomowe

Nadążając za potrzebami rynku pracy, szkoły wyższe uruchamiają kierunki informatyczne i modyfikują programy nauczania, wprowadzając do tradycyjnych kierunków specjalności „informatyczne" [1, 4]. Przykładem takich działań jest uruchomienie na Wydziale Elektrycznym Politechniki Lubelskiej w ramach kierunku Elektrotechnika specjalności *Inżynierskie zastosowania informatyki* (IZI) o dwóch specjalizacjach: *Komputerowe wspomaganie projektowania i Komputerowe systemy automatyki.* Przedmioty informatyczne specjalności IZI obejmują 1245 godzin dydaktycznych, w tym około 300 godzin specjalizacyjnych [4]. Pewną cechą wspólną specjalności „informatycznych" w wyższych szkołach jest obecność przedmiotów niekoniecznie związanych z podstawowym kierunkiem kształcenia, ale za to podnoszących pozycję absolwenta na rynku pracy. Ich program obejmuje przede wszystkim projektowanie systemów informatycznych (głównie wspomagających zarządzanie przedsiębiorstwem), bazy danych i sieci komputerowe (w aspektach projektowania, budowy i administrowania, a także Internetu).

„Informatyczne" specjalności w tradycyjnych kierunkach kształcą użytkowników systemów informatycznych w różnych dziedzinach. Ze względu na długi cykl kształcenia efekty tych działań są widoczne bardzo późno.

Wyjściem z sytuacji są studia podyplomowe, które ze względu na krótki cykl kształcenia (zwykle 2–3 semestry) są doskonałą metodą uzupełnienia wykształcenia bądź też nawet zmiany jego profilu. Studia podyplomowe z reguły są płatne i prawdopodobnie dzięki temu rozwijają się w błyskawicznym tempie [5]. Przykładem może być pięciokrotny wzrost liczby studiów podyplomowych w Politechnice Lubelskiej w ciągu ostatnich dwóch lat. Popularność studiów podyplomowych wynika z przemian w gospodarce Polski i wiążących się z nimi zmian w strukturze zatrudnienia, a także, w przypadku studiów informatycznych, z szybkim rozwojem technologii. Pierwsza przyczyna tej popularności prawdopodobnie będzie z czasem tracić znaczenie. Druga natomiast wydaje się zapewniać stałą koniunkturę studiom podyplomowym, związanym z informatyką.

Fakt dużej popularności studiów podyplomowych może cieszyć, ale ponieważ uczą się na nich ludzie młodzi (np. ponad 44% w wieku poniżej 30 lat – rys. 1), w tym nawet studenci, powinien wywołać pewną refleksję. Czy szkoły wyższe przypadkiem nie „produkują" absolwentów, którzy by dostać pracę muszą się natychmiast przekwalifikować lub dokształcać? Czy nie należałoby uaktualnić programów i ograniczyć liczbę słuchaczy niektórych kierunków kształcenia w szkołach wyższych? Czy stać nas na bezrobotnych magistrów?



Rys. 1. Struktura wiekowa słuchaczy podyplomowego studium organizowanego przez Politechnikę Lubelską i Uniwersytet Illinois (oprac. własne)

Studia podyplomowe są mimo wszystko **wyjątkowo aktrakcyjną formą** podnoszenia kwalifikacji. Gwarantują zdobycie w miarę szerokiej wiedzy, są prowadzone przez doświadczonych wykładowców i stosunkowo mało kosztują (jeśli się uwzględni liczbę godzin zajęć, intensywność nauczania i poziom merytoryczny). Niski koszt studiów podyplomowych uwypukla się w porównaniu ze specjalistycznymi szkoleniami, organizowanymi przez różne firmy informatyczne. Nie należą do wyjątków koszty takich szkoleń w wysokości 500 i więcej złotych dziennie. W studiach podyplomowych pewnym mankamentem jest konieczność uzyskiwania przez słuchaczy zaliczeń i zdawania egzaminów, a niekiedy nawet przygotowanie pracy dyplomowej. Takie formy kontroli nie istnieją przy szkoleniach. W związku z tym inna jest waga

świadectwa ukończenia studiów podyplomowych, organizowanych przez uczelnie, a inna zaświadczenia o ukończeniu kursów czy szkoleń.

Kursy i szkolenia

W modelu informatycznej edukacji społeczeństwa swoje miejsce powinien znaleźć szeroki system szkoleń komputerowych. Kursy takie powinny dawać wiedzę na temat najnowszego oprogramowania i sprzętu informatycznego. Organizacja krótkotrwałych i specjalizowanych szkoleń charakteryzuje się małą inercyjnością. Można je uruchamiać prawie natychmiast reagując na potrzeby rynku, zmianę technologii, pojawienie się nowej wersji programu itp. Nie gwarantują one zdobycia kompleksowej wiedzy w określonej dziedzinie i nie mogą zastąpić studiów. Są natomiast ich doskonałym uzupełnieniem i naturalną formą ustawicznego kształcenia i podnoszenia umiejętności zawodowych w zakresie informatyki.

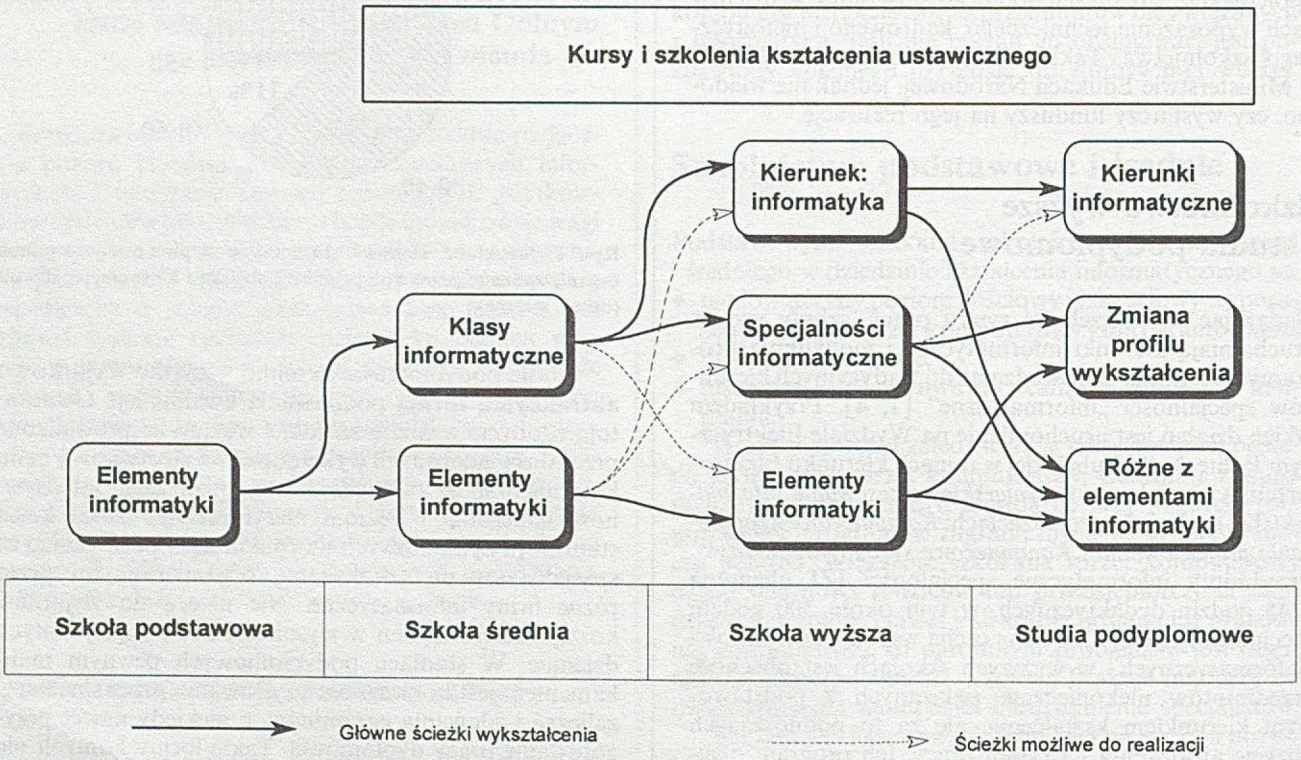
Pozostaje oczywiście problem atestacji i weryfikacji kursów. Dzisiaj każdy, kto ma komputer i jakąś wiedzę o oprogramowaniu, może otworzyć kolejne *Bardzo Ważne Informatyczne Centrum Szkoleniowe*. Nauczanie informatyki jest jedną z nielicznych dziedzin nauczania, w których innych można szkolić dłużej niż samemu pobierało się naukę [8]. Nauczycielami (przynajmniej tak im się wydaje) w tej dziedzinie stają się ludzie, którzy trochę postukali jakiś czas w klawisze lub poklikali myszą. I czego potem nauczają na różnego typu kursach? Obsługi konkretnego programu, układów menu, sekwencji klawiszy i znaczeń przycisków. Są w tym sprawni, a zatem to świetni fachowcy – myślą sobie słuchacze kursów. Znacznie sprawniejsi od nauczyciela, który kiedyś z mozołem objaśniał diagramy składniowe, ideę rekurencji, zależność wielkości błędu obliczeń od parametrów algorytmu czy też zasady poprawnej edycji tekstów. I w ten sposób w informatyce mamy „wspaniały" stosunek uprawiających informatykę do specjalistów: około 1000 do 1 [8].

V Jubileuszowe Warsztaty Mikrokomputerowe w Kazimierzu

Warsztaty Mikrokomputerowe Politechniki Lubelskiej są regionalną konferencją o dość luźnej formule, poświęcaną każdorazowo innym problemom. W tym roku V Jubileuszowe Warsztaty zostały zorganizowane przez Politechnikę Lubelską (Zakład Systemów Informatycznych Katedry Zarządzania i Katedrę Informatyki) i Lubelskie Koło PTI przy współdziałaniu Kuratorium Oświaty w Lublinie. Były one jakby preludium do XII Konferencji „Informatyka w Szkole", która w tym roku odbędzie się w Lublinie w dniach 11–14 września.

Program V Warsztatów poświęcony był problemom kształcenia informatycznego na bardzo różnych poziomach: począwszy od szkół podstawowych, poprzez szkolnictwo średnie i wyższe, aż po różne formy kształcenia podyplomowego i ustawicznego ze szczególnym uwzględnieniem uwarunkowań Makroregionu Środkowo-Wschodniego.

Warsztaty rozpoczęły się od identyfikacji potrzeb i określenia modelu użytkownika systemów informatycznych. W tej materii wiele wyjaśniły wystąpienia zaproszonych gości z ośrodków przemysłowych Lubelszczyzny. Wypowiedzi szefów informatyki dużych zakładów, banków i niewielkich firm usługowych koncentrowały się na wymaganiach umiejętności i sprawności w posługiwaniu się technikami informatycznymi w połączeniu z dużą wiedzą teoretyczną, która umożliwi pracownikowi bezboleśnie przejść na inny system operacyjny, program użytkowy lub sprzęt.



Rys. 2. Schemat modelu edukacji informatycznej ze ścieżkami kształcenia

Na kolejnych sesjach Warsztatów rozpatrywano problemy edukacji informatycznej na coraz wyższych szczeblach nauczania. Uzupełnieniem Warsztatów były konferencje metodyczne Kuratorium Oświaty w Lublinie przeznaczone dla nauczycieli informatyki w szkołach podstawowych i średnich.

Model edukacji informatycznej

Szeroko rozumiany model edukacji informatycznej społeczeństwa powinien być w miarę spójnym i jednocześnie elastycznym systemem kształcenia użytkowników informatyki na wszystkich szczeblach edukacji (rys. 2). Powinien on także określać system ustawicznego kształcenia.

Charakterystyczną cechą modelu edukacji informatycznej jest obecność przedmiotu o nazwie *Elementy Informatyki*. Przedmiot ten jest często inaczej nazywany. Mogą to być *Podstawy Informatyki, Informatyka Stosowana, Informatyka Techniczna,* trącące myszką *Elektroniczne Przetwarzanie Danych* lub też po prostu *Informatyka.* Niezależnie od nazwy przedmiot ten jest umiejscowiony wśród takich przedmiotów, jak matematyka czy język obcy. Powinien więc zostać uznany za **przedmiot ogólnokształcący** i jego nauczaniem powinni być objęci wszyscy uczniowie. Z modelu wynika także konieczność opracowania **spójnego programu tego przedmiotu począwszy od szkoły podstawowej aż po wyższą.** W tej materii jest wymagana zatem ścisła współpraca środowiska oświaty ze środowiskiem akademickim. Z treści przedmiotu Elementy Informatyki na poszczególnych szczeblach kształcenia będą wynikać bowiem dalsze potrzeby: kadrowe, sprzętowe, oprogramowania, podręczników itp.

* * *

Przedstawiony model odwzorowuje tendencje w kształceniu informatycznym. W praktyce muszą w modelu tym znaleźć swoje miejsce takie formy kształcenia, jak: zawodowe szkoły informatyczne, szkoły pomaturalne i studia licencjackie. Najbardziej pilną potrzebą jest jednak uporządkowanie i ujednolicenie nauczania elementów informatyki.

LITERATURA

[1] Bolkowski S., Osowski S.: Komputerowe Metody w Elektrotechnice – kierunek informatyczny na Wydziale Elektrycznym Politechniki Warszawskiej. Informatyka Stosowana, nr 1/1995, s. 73–85

[2] Miłosz E., Miłosz M., Perkowski R., Smołka J. W.: Zaoczne kształcenie nauczycieli elementów informatyki. X Konferencja „Informatyka w Szkole", Toruń, 1994, str. 55–60

[3] Miłosz E., Smołka J.: Podyplomowe informatyczne kształcenie nauczycieli w Politechnice Lubelskiej. Informatyka Stosowana, Zeszyt Naukowy nr 1, nr S1/1996, s. 125–132

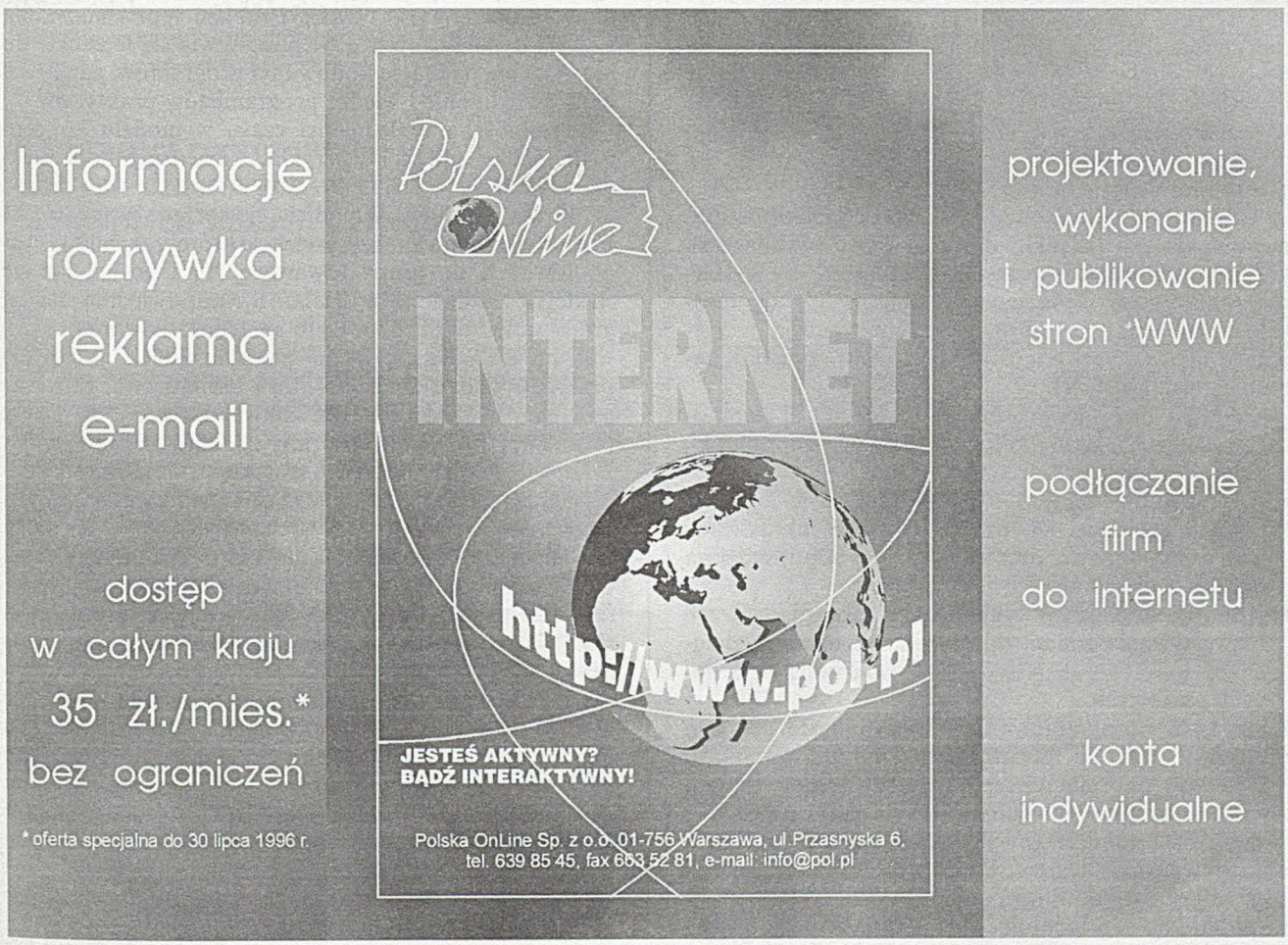
[4] Miłosz M.: Inżynierskie zastosowania informatyki – nowa specjalność na Wydziale Elektrycznym Politechniki Lubelskiej. Informatyka Stosowana, nr 2/1994, s. 91–99

[5] Miłosz M.: Studia podyplomowe – moda czy konieczność? Informatyka Stosowana, Zeszyt Naukowy nr 1, nr S1/1996, s. 147–153

[6] Ramowy program działań w zakresie edukacji informatycznej w oświacie w latach 1996–1997 (projekt). MEN, Warszawa, luty 1996

[7] Raport I Kongresu Informatyki Polskiej: Strategia Rozwoju Informatyki w Polsce – Stan, Perspektywy, Zalecenia. Poznań, 1994

[8] Sysło M.: Zmierzch nauczania? Zbędny nauczyciel? CHIP, nr 3/1996, s. 34.



Zagadnienia reprezentacji czasu w klasycznych systemach baz danych

Krzysztof Kania
Jerzy Gołuchowski
Katowice

Mimo iż czas jest jedną z najważniejszych własności otaczającej nas rzeczywistości, a dynamiczna natura zdarzeń jest niekwestionowaną cechą naszego otoczenia, zagadnienia ujęcia czasu w modelach konceptualnych rzeczywistości oraz w bazach danych stanowią kwestię nadal niedostatecznie rozwiązaną w metodologii tworzenia inteligentnych systemów informatycznych zarządzania (ISIZ). Taki stan rzeczy wynika nie tylko z dozwolonego upraszczania rzeczywistości i ujmowania jej w sposób statyczny, ale także z niedostatków stosowanego aparatu metodologicznego oraz narzędzi programowych.

Po wielu dyskusjach nad problemem reprezentacji czasu w systemach baz danych (SBD) przyjęło się ujmować czas jako atrybut obiektów zarówno na poziomie modelu konceptualnego, jak i na poziomie logicznym (jako atrybuty w relacji). W prostych zastosowaniach takie ujęcie może być zadowalające, aczkolwiek nie jest wolne od ograniczeń, których często nawet nie dostrzegają konstruktorzy aplikacji, przywykli do narzuconego im sposobu myślenia o czasie.

Gdy usiłuje się budować inteligentne systemy wspomagania decyzji (ISWD), dotychczasowe ujęcia czasu w systemach informatycznych okazują się niewystarczające. Z tego względu zauważa się powrót zainteresowania problematyką czasu, jego reprezentacją w systemach informatycznych (SI) oraz modelowaniem dynamicznej rzeczywistości.

Badania nad ujęciem informacji czasowej (temporalnej) rozwijają się w kilku dziedzinach informatyki, w szczególności w obszarze [1, 3]:

- systemów baz danych – podejmowane w aspekcie potrzeb gromadzenia, przechowywania, aktualizacji informacji czasowej oraz uzyskiwania odpowiedzi na pytania z takich baz,
- sztucznej inteligencji – analizujące czas dla potrzeb rozwiązywania zadań, takich jak: planowanie, diagnozowanie, tworzenie scenariuszy na podstawie danych czasowych,
- inżynierii oprogramowania – podejmujące problem czasu w zakresie modelowania i formalnego opisu systemów informatycznych ujmujących dane (w tym również programy) w ich czasowym kontekście.

Klasyczne modele konceptualne ograniczają się do odwzorowywania elementów niezmiennych w rzeczywistości: obiektów, powiązań między nimi oraz więzów integralności, reprezentujących pewne prawidłowości w modelowanej dziedzinie. Opisując dynamikę na etapie modelowania konceptualnego projektanci SI – do tej pory – najczęściej ograniczają się do uzupełnienia wytworzonego schematu o werbalny opis wybranych powiązań pomiędzy zdarzeniami występującymi w modelowanym systemie oraz do wskazania powiązań między tymi zdarzeniami a procedurami ich obsługi. Ani rodzaju zmian rzeczywistości, ani ich konsekwencji dla zapisu danych nie odnotowuje się zatem formalnie w projekcie.

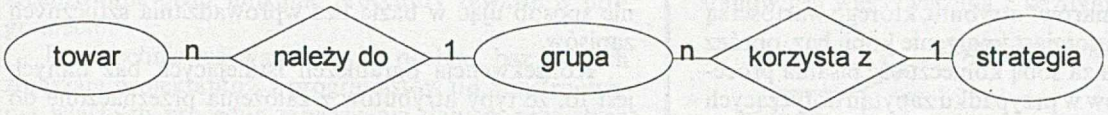
Trudności ujęcia czasu w klasycznych konceptualnych modelach danych

W klasycznych SI opartych na bazie danych czas traktuje się w sposób uproszczony, sprowadzając go wyłącznie do atrybutu reprezentowanego obiektu. W najlepszym przypadku wartości atrybutów zmieniających się w czasie są ujmowane w bazie danych za pomocą odrębnego typu danych (np. DATE, TIME). Problem reprezentowania zjawisk dynamicznych w SI pozostawia się w istocie do rozwiązania na etapie kodowania programów, zdając się na doświadczenie i wiedzę programistów-praktyków.

Trudności uwzględnienia czasu w modelu konceptualnym ilustruje prosty przykład z dziedziny modelowania sprzedaży.

Firma handlowa Z podjęła się sprzedaży towaru X. Przeprowadzono analizę, w efekcie której zakwalifikowano towar X do grupy towarowej T_1 i posługując się strategią zestawioną z działań S_1, odpowiednią dla tej grupy towarowej, rozpoczęto sprzedaż. Towar sprzedawał się słabo, dlatego po pewnym czasie ponownie przeprowadzono analizę rynku i stwierdzono, że towar powinien być od początku zakwalifikowany do grupy towarowej T_2 i rozprowadzany zgodnie ze strategią S_2. Ponieważ podjęto decyzję o zmianie strategii, odpowiednie informacje powinny zostać opisane i zapisane w systemie.

Schematy: konceptualny (oparty na metodologii OAR) i logiczny (relacyjny) przedstawionej rzeczywistości, jakie tworzy się w tradycyjnym ujęciu, przedstawia rys. 1. Jak widać, model konceptualny w takiej postaci nie zawiera żadnych informacji o możliwości zmiany strategii marketingowej w czasie sprzedaży towaru, a nawet nie wskazuje na potrzebę uwzględnienia czasu w modelu logicznym SI. Jeśli ponadto przyjmie się tradycyjne ujęcie bazy danych, w którym zakłada się, że przedstawia ona stan bieżący, to nowe wartości zostaną wpisane w miejsce starych i utraci się dane o nieprawidłowym rozpoznaniu w przeszłości oraz informację o tym, że towar był sprzedawany według strategii S_1. Tym samym utraci się również możliwość analizy wpływu poprzednich działań



Towar(id_towaru, nazwa, id_grupy,...)
Grupa(id_grupy, nazwa, id_strategii, ...)
Strategia(id_strategii, opis_strategii,...)

Towar

id_towaru	nazwa	id_grupy
1	X	1

Grupa

id_grupy	nazwa	id_strategii
1	t1	s1

Strategia

id_strategii	opis_strategii
s1	opis działań

Rys. 1. Fragment modelu konceptualnego i logicznego systemu sprzedaży

na obecne. Ponadto, jeśli nowe dane nie będą zawierały zapisów o dokonanej zmianie strategii dla tego towaru, to można je mylnie zinterpretować jako przekwalifikowanie towaru z jednej grupy towarowej do innej, co również może doprowadzić do błędnych interpretacji i wniosków.

Przedstawiony przykład wskazuje na znaczenie i trudności prawidłowego zapisu danych czasowych, już w początkowej fazie tworzenia SI. Zadaniem projektanta systemu jest więc wnikliwe rozważenie sposobu ujęcia dynamiki zjawisk oraz odpowiednie ich przedstawienie w sformalizowanej wersji projektu.

Konsekwencją rozwiązywania problemów reprezentacji czasu w klasycznych modelach SI, dopiero przez programistów, jest wiele poważnych problemów implementacyjnych. Są one rozwiązywane różnymi sposobami. Następstwem tego jest także brak jednolitych mechanizmów działania na danych, których historia musi być przechowana w bazie danych. Niejednolitość rozwiązań spotyka się nawet w tym samym SI.

Trudności reprezentacji danych czasowych w tradycyjnych, relacyjnych bazach danych

Podstawowym typem baz danych stosowanym w tradycyjnych SI są tzw. bazy (relacje) chwilowe (ang. *snapshot relation*). Ujmują one tylko bieżący stan rzeczywistości, tak jak klatka filmu przedstawia akcję uchwyconą (zatrzymaną) w ruchu. Takie bazy danych mają niewątpliwie ograniczoną przydatność: jedynie wspierają podejmowanie decyzji operacyjnych. Praktycznie relacje o takiej strukturze mogą mieć zastosowanie w sytuacjach, gdy użytkownika interesuje jedynie bieżący stan wartości atrybutów. Pozwalają one na uzyskanie odpowiedzi na jeden typ pytania: *Jaka informacja o danym atrybucie została aktualnie wpisana do bazy?* Istotną cechą takich baz jest zastępowanie wartości nieaktualnych nowszymi, przy czym dane sprzed aktualizacji są bezpowrotnie stracone [2].

Jeśli zachodzi potrzeba zapamiętania zmian w bazie, to programista staje przed problemem rozstrzygnięcia we własnym zakresie:

- jakie zastosować struktury danych,
- jakie skonstruować funkcje obsługi stworzonych struktur.

W tradycyjnych SZBD (Systemy Zarządzania Bazą Danych) oprócz standardowych typów danych (np. znakowych i numerycznych) najczęściej do dyspozycji programisty są:

- typy danych umożliwiające reprezentowanie wartości opisujących czas (DATE, TIME),
- zbiór podstawowych funkcji operujących na tych typach danych (porównywanie, obliczanie dat itp.).

Ujęcie czasu w tradycyjnych bazach ogranicza się zatem do prostych atrybutów, których znaczenie jest definiowane w relacji (lub zbiorze kopii relacji) przez programistów. Taki sposób posługiwania się informacją czasową określa się jako indywidualny sposób definiowania czasu przez użytkownika (ang. *user-defined time*).

O ile chwilowe relacyjne bazy danych zadowalają np. personel obsługujący magazyn, taka konstrukcja okazuje się niewystarczająca do udzielenia odpowiedzi na wiele podstawowych pytań stawianych przez kierownictwo przedsiębiorstw w procesie podejmowania decyzji gospodarczych. Reprezentacja informacji przyjęta w tego rodzaju modelu danych okazuje się nieadekwatna do wymagań obliczeniowych rozwiązywanych problemów analizy i podejmowania decyzji. Przedstawimy to na przykładowej typowej relacji chwilowej (Tabela 1), opisującej stan wybranych towarów w hurtowni artykułów gospodarstwa domowego.

Tabela 1. Relacja: Towar

Towar	Cena	Dostawca
łyżki	20	A
widelce	30	A
noże	40	B

Aby ominąć ograniczenia znormalizowanych, relacyjnych BD programiści uciekają się do – w pewnym sensie – nienaturalnych rozwiązań:

- dodając do każdego atrybutu zmiennego w czasie (lub relacji) dodatkowe pole (np. do pola cena – datę zmiany ceny) lub
- tworząc kopie archiwalne relacji (np. sprzedaż w miesiącu X).

Dla wyrażenia wiedzy o wartości danego atrybutu zmiennego w czasie (np. ceny) programiści rozbudowują relacje o odrębne, sztuczne atrybuty określające czas zmiany wartości interesujących użytkownika atrybutów, gdyż nie mogą ująć inaczej w bazie zmian wartości tych atrybutów. Sztuczność tego rozwiązania polega właśnie na tym, że dodatkowy atrybut czasowy jest bytem samodzielnym w relacji, podczas gdy faktycznie jest on

tylko jednym z „wymiarów" atrybutu, którego wartości są zmienne w czasie. Natomiast tworzenie kopii baz, oprócz redundancji, pociąga za sobą konieczność pisania procedur scalania archiwów w przypadku zapytań dotyczących stanu bieżącego i przeszłego.

Tabela 2. Relacja: Towar1

Towar	Cena	Data_c	Dostawca	Data_d
łyżki	15	01.03.94	B	20.02.94
łyżki	20	01.05.94	A	20.05.94
łyżki	20	01.06.94	B	01.06.94
widelce	30	01.05.94	C	25.05.94
widelce	30	10.03.94	A	25.06.94
noże	40	01.05.94	B	01.05.94

Relacja Towar1 przedstawiona w Tabeli 2 jest przykładem typowego opisu danych zmiennych w czasie, jaki można spotkać w wielu funkcjonujących obecnie SI. Ilustruje ona sposób, w jaki programista usiłuje uporać się z zapisem wartości tego rodzaju atrybutów w tradycyjnych SZBD. Podkreślić należy, że za odpowiednią reprezentację informacji czasowej całkowitą odpowiedzialność ponosi obecnie programista. SZBD nie dostarcza bowiem ani żadnych mechanizmów pozwalających na wyróżnienie atrybutów zmiennych w czasie, ani narzędzi manipulowania danymi temporalnymi.

Oprócz pracochłonności implementacji (np. budowa procedur odzyskiwania informacji) wadą tego typu rozwiązań jest także to, że niejasna pozostaje kwestia interpretacji zapisu wartości atrybutów czasowych. Zapis ten z reguły traktuje się jako moment, w którym dana wielkość powstała w realnym świecie. Zakładając, że w bazie obowiązuje zasada stanowiąca, że początek następnego okresu czasu jest jednocześnie końcem poprzedniego, interpretacja pierwszych zapisów z relacji Towar1 jest następująca:

Cena 15 na łyżki obowiązywała od 01.03.94 do 30.04.94 (01.05.94 uległa zmianie).

Dostawcą łyżek między 20.05.94 a 30.05.94 była firma A (od 01.06.94 firma B).

Warto zauważyć, że konieczne jest przyjęcie także innych założeń, np. że wartości z ostatniej krotki, dotyczącej danego towaru, są aktualne do chwili bieżącej. Określenie natomiast, że towar kiedyś znajdował się w magazynie, lecz obecnie go nie ma lub nie jest określony jego dostawca, zmusza do utrzymywania zapisów z wartościami pustymi (NULL). Inną cechą takiego, często stosowanego obecnie rozwiązania jest założenie ciągłości występowania wartości i na przykład faktu, że w magazynie były materiały z nie ustaloną ceną (faktura w drodze), nie sposób ująć w bazie bez wprowadzania sztucznych zapisów.

Konsekwencją ograniczeń istniejących baz danych jest to, że typy atrybutów z założenia przeznaczone do przechowywania takich wartości, jak data urodzenia, data zatrudnienia itp. muszą być również wykorzystywane do przedstawienia historii zmian wartości atrybutu. Jest to rozwiązanie nienaturalne i niepraktyczne z kilku powodów, m.in.:

- powoduje brak jednoznacznego i jednolitego podejścia do rozwiązania problemu przedstawiania dynamiki obiektów,
- zmusza programistów do tworzenia ad hoc własnych rozwiązań,
- nie pozwala na uwzględnienie czasu w sposób naturalny dla użytkownika,
- nie ujmuje czasu jako wymiaru rzeczywistości, a tylko jako jeden z wielu atrybutów,
- pozwala na różne interpretacje i reprezentacje czasu.

Pomimo swych mankamentów tradycyjny sposób zapisywania informacji o czasie jest często spotykany w praktyce i zadowala on wielu użytkowników. Nierzadko jednak nie wystarcza on do precyzyjnego opisu złożonej rzeczywistości, a także do efektywnego uzyskiwania informacji z baz danych. Jednym ze sposobów przezwyciężenia ograniczeń tradycyjnego modelu danych jest wprowadzenie nowych, bardziej złożonych typów danych. Przedstawia je Tabela 3. Pożądanym rozwiązaniem byłoby, aby typy danych wymienione w tej tabeli były wbudowane w SZBD, a nie jak dotychczas musiały być konstruowane i obsługiwane na poziomie aplikacji. Zmiana ta powinna zaowocować podniesieniem jakości tworzonych SI.

W niektórych dziedzinach, ze względu na konieczność dokładnego opisu historii rzeczywistości lub w celu zachowania logiki bazy, występują ponadto dane opisujące kolejne „inkarnacje" obiektów. Przykładem użycia takich danych może być opis pracownika wielokrotnie przyjmowanego i zwalnianego z tego samego zakładu pracy.

Czas w istniejących językach zapytań

Reprezentacja danych nie jest celem samym w sobie, lecz jest podporządkowana celom przetwarzania danych. Brak standardowych rozwiązań w zakresie zapisu danych czasowych w bazach danych ma swoje negatywne konsekwencje również w braku odpowiednich mechanizmów manipulowania danymi w SZBD oraz w budowie języków zapytań. Ponieważ używane języki zapytań są stworzone do obsługi chwilowych baz danych, nie mają wbudowanych klauzul pozwalających na zadawanie pytań o dynamikę zjawisk. Każde takie pytanie musi być

Tabela 3. Typy danych w odniesieniu do czasu

Rodzaj wartości	Opis	Przykład
identyfikatory	niezmienne w czasie identyfikujące obiekty	nr PESEL
niezmienne w czasie	stałe, niezależne od czasu	miejsce urodzenia, imiona rodziców
zmienne w czasie	dane, które mogą dezaktualizować się	adres zamieszkania, stanowisko
określające czas	dane określające czas	data urodzenia, data zatrudnienia
nakładające się	dane opisujące kolejno nabywane własności obiektu	znajomość języków, kursy zawodowe
czasowo określone	dane opisujące własności, które obiekt nabędzie lub utraci w przyszłości	czasowe uprawnienia, próbny okres zatrudnienia

zrealizowane przez unikalne procedury zawarte w programach.

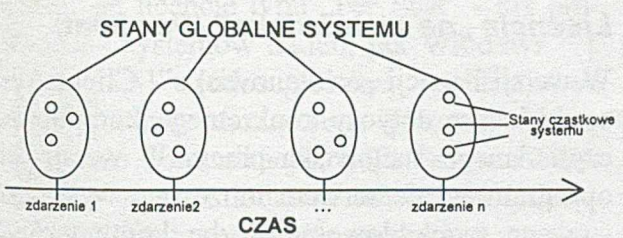
Powszechnie używane języki obsługi baz danych zmuszają projektantów i programistów do konstruowania własnych procedur zapisywania danych przeszłych oraz odzyskiwania danych z utworzonych archiwów. W rezultacie rozwiązania te są najczęściej mało elastyczne oraz wymagają ciągłych poprawek i ulepszeń albo ograniczają użytkownika do zadawania z góry określonych pytań, zawartych w aplikacji. Rozbudowa specyficznych procedur bardzo utrudnia integrację systemów, ponieważ łączone moduły mają różne sposoby manipulacji takimi danymi, a często także różną reprezentację. Również nieunikniony rozwój systemu utrudnia odzyskiwanie danych przeszłych, ponieważ zmiana struktury baz (nazwy, rozmiary pól, powiązania, zmiany kluczy), pociąga za sobą konieczność zmian sposobów archiwacji i odzysku danych.

Przeniesienie możliwości konstruowania zapytań, dotyczących danych czasowych, na poziom języka, a manipulowanie takimi danymi chociażby częściowo na poziomie SZBD (a nie tylko i wyłącznie programu) poszerzyłoby znacznie możliwości systemów w zakresie reprezentowania dynamicznej rzeczywistości, zwiększyłoby wydajność pracy informatyków, a także ułatwiłoby modyfikacje, wprowadzane przez użytkowników systemów. Systemy spełniające takie wymagania nazywamy systemami czasowych baz danych (ang. *temporal databases*). Pełne wykorzystanie możliwości, jakie dają takie bazy danych jest jednak możliwe jedynie wtedy, gdy począwszy od etapu modelowania przez konstrukcję odpowiednich baz konsekwentnie jest brany pod uwagę aspekt czasu.

Warianty reprezentacji czasu w informatycznych systemach zarządzania

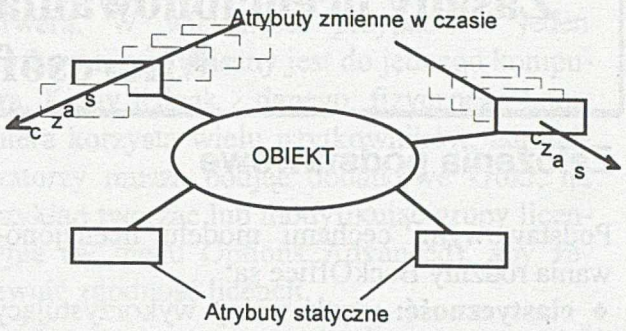
W literaturze przedmiotu spotyka się dwa zasadnicze modele, które starają się pokonać trudności ujęcia dynamiki świata w systemach baz danych [5]. Różnią się one sposobem ujęcia czasu w modelu.

Pierwsze z nich zakłada, że czas jest czynnikiem istniejącym niezależnie od poszczególnych obiektów, istnieje zatem jako wymiar modelowanej rzeczywistości. Na osi czasu są „odkładane" zdarzenia, które wpływają na stan systemu. Z każdą pozycją na osi czasu jest związany zbiór jego stanów. Wiedząc, jakie zdarzenia miały lub będą miały miejsce, możemy odtworzyć lub przewidzieć stan, w jakim znajdował się system w badanym czasie. Takie ujęcie jest typowe dla ogólnej teorii systemów [4]. Historia obiektu jest więc wyznaczona przez zbiór kolejnych zdarzeń (rys. 2.).



Rys. 2. Zdarzeniowa interpretacja czasu

W drugim modelu czas traktowany jest jako składnik opisu poszczególnych atrybutów obiektu. Historia jego stanów jest więc związana z każdym atrybutem (rys 3.).



Rys. 3. Czas jako historia atrybutów obiektu

Niezależnie od przyjętego sposobu ujęcia czasu, tak na poziomie konceptualnym, jak i na poziomie logicznym, czas przestaje być traktowany tylko jako jeden z równorzędnych atrybutów i można posługiwać się nim jako dodatkowym wymiarem możliwym do wbudowania tak w model rzeczywistości, jak i jego implementację do bazy danych. Aby mogło to jednak nastąpić, należy także rozstrzygnąć problemy związane z reprezentacją i interpretacją czasu oraz określić wpływ podjętych decyzji na system komputerowy.

* * *

W artykule usiłowano pokazać, że dotychczasowe ujęcie dynamiki rzeczywistości i czasu w SBD – w przypadku budowy bardziej złożonych aplikacji – jest zbyt restrykcyjne i zawodne. Wyodrębnienie atrybutów zmiennych w czasie, obok dotychczas stosowanych atrybutów opisujących czas, pozwala projektantowi myśleć mniej zawile o modelowanej rzeczywistości. Zwalnia go także od uciążliwej samokontroli, przekazując kontrolę nad tymi atrybutami SZBD.

Dla tych, których w modelowaniu rzeczywistości zadawala traktowanie czasu tylko jako jednego z atrybutów obiektu, nowe technologie systemów baz danych niosą odpowiedź na pytanie jak zapisywać dane z przeszłości i przyszłości tak, aby łatwo można było zadawać pytania o wartości atrybutów czasowych. Dla tych, którzy postrzegają czas jako wymiar modelowanej rzeczywistości, który powinien być formalnie i konsekwentnie ujęty w SI, modelowanie danych czasowych i takież bazy danych są sprawnym narzędziem umożliwiającym reprezentację czasu.

W kolejnym artykule przedstawimy propozycję ujecia czasu w modelu konceptualnym SI.

LITERATURA

[1] Clifford J., Rao A.: A Simple, General Structure for Temporal Domains. w: Temporal Aspects in Information Systems. Rolland C., Bodart F., Leonard M. (Eds.) North-Holland, Amsterdam, 1988

[2] Ling D., Bell D.: Taxonomy of Time Models in Databases., Information and Software Technology, vol. 32, no. 8, 1990

[3] Maiocchi R., Percini B.: Temporal Data Management Systems: A Comparative View. IEEE Transaction on Knowledge and Data Engineering, vol. 3, no. 4. 1991

[4] Sadowski W.: Podstawy ogólnej teorii systemów. PWN, Warszawa, 1978

[5] Theodoulidis C., Loucopoulos P.: The time dimension in conceptual modelling. Information Systems, vol. 16, no. 3, 1991.

Kontakt z autorem:
email: {kania|goluch@legato.ae.katowice.pl}

Zasady licencjonowania i zakupu pakietów rodziny Microsoft BackOffice

Założenia podstawowe

Podstawowymi cechami modelu licencjonowania rodziny BackOffice są:

- **elastyczność:** użytkownicy wykorzystujący oprogramowanie na wiele różnych sposobów mogą wybrać takie opcje modelu licencjonowania, jakie najbardziej pasują do ich potrzeb;
- **przewidywalność:** licencjonowanie Microsoftu płynnie podąża za rozbudową systemu użytkownika, dodającego nowe serwery i stacje robocze. W modelu tym nie ma ukrytych kosztów czy też progowych skoków cen, kiedy rosną potrzeby użytkownika;
- **narzędzia administratora:** Windows NT 3.51 Server zawiera narzędzia zarządzania licencjami, pozwalające kontrolować w ramach organizacji wykorzystywanie licencjonowanego oprogramowania;
- **konkurencyjność cenowa:** każdy element z rodziny Microsoft BackOffice ma atrakcyjną cenę w porównaniu do produktów konkurencyjnych, a jako cały pakiet Microsoft BackOffice daje dalsze korzyści cenowe.

W skład rodziny produktów Microsoft BackOffice, której dotyczy poniższy model, wchodzą: Windows NT Server, SNA Server, SQL Server, Systems Management Server oraz Microsoft Mail (w przyszłości Microsoft Exchange).

Oddzielne licencje dla serwera oraz licencje dostępu dla stacji roboczej

W modelu licencjonowania Microsoftu licencje dostępu dla stacji roboczej nabywa się odrębnie od licencji serwera. Użytkownicy kupują tylko taką liczbę licencji typu Client Access License, jakiej potrzebują - po jednej dla każdego komputera, z którego wymagany jest dostęp do oprogramowania serwera. Licencja typu Client Access License to dokument o mocy prawnej, potwierdzający prawo użytkownika do dostępu do usług oprogramowania z danego serwera. Warto zauważyć, że licencja typu Client Access wymagana jest niezależnie od tego, czy używamy oprogramowania klienta dostarczonego przez Microsoft czy też przez niezależne firmy trzecie. Na przykład jeśli komputer działa pod kontrolą Windows for Workgroups albo Windows NT Workstation bądź też dowolnego innego oprogramowania obsługiwanego przez Windows NT Server, wymagana jest oddzielna licencja o nazwie Client Access License for Windows NT Server.

Pełna sieć BackOffice wymaga co najmniej jednej licencji na oprogramowanie serwera oraz tylu licencji Client Access License, ile komputerów podłączonych jest do serwera. Na przykład pełna sieć 10 stacji wymaga 1 licencji na serwer oraz 10 licencji Client Access.

Gdy użytkownik chce dodać do sieci dodatkowy serwer, musi wykupić dodatkową licencję serwerową, gdy natomiast zwiększa ilość stacji podłączonych do serwera, dokupuje tylko dodatkowe licencje dostępu dla stacji, niezależnie od tego, jak wiele serwerów obsługuje daną sieć.

Dwie opcje licencjonowania

W ramach licencjonowania rodziny BackOffice Microsoft oferuje dwie podstawowe opcje, do wyboru przez nabywcę, w zależności od tego, która z nich najbardziej odpowiada jego potrzebom. Nie ma nawet potrzeby powiadamiania Microsoftu, którą z opcji wybrano. W obu wersjach licencja na serwer kupowana jest oddzielnie od licencji Client Access.

Licencja „na stanowisko" (Per Seat)

W wersji licencji „na stanowisko" Client Access License dotyczy konkretnego komputera, czyli danego stanowiska pracy. W wersji tej oprogramowanie serwera może być wykorzystywane przez dowolną liczbę komputerów, jeśli tylko każdy z nich ma odpowiednią Client Access License. Na przykład komputer, na który wykupiono licencję na dostęp do Win-

dows NT Server, może wykorzystywać ten dostęp do dowolnej liczby serwerów Windows NT Server znajdujących się w dowolnym miejscu firmy (i połączonych, np. za pomocą łącz modemowych, X.25, ISDN itp). Tego rodzaju wersja licencji dostępna jest dla wszystkich produktów rodziny Microsoft BackOffice.

Licencja „na serwer" (Per Server)

W wersji licencji „na serwer" Client Access License dotyczy konkretnego serwera.
W wersji tej można stosować tzw. licencjonowanie równoległego korzystania z oprogramowania serwera. Nabywca, który zdecyduje się na ten model licencjonowania, musi mieć co najmniej tyle licencji typu Client Access dla danego serwera, ile wynosi maksymalna liczba użytkowników, którzy podłączą się do tego serwera w danej chwili. W wersji tej nabywca określa liczbę licencji Client Access, która dotyczyć będzie danego serwera w momencie instalacji oprogramowania serwera. Taka wersja licencji dostępna jest dla oprogramowania Windows NT Server, SQL Server, SNA Server oraz Microsoft Exchange (po jego udostępnieniu), nie jest natomiast dostępna dla serwera Systems Management Server oraz w wersji BackOffice Client Access License.

Zarządzanie licencjami

System Windows NT Server 3.51 dostarcza nowe narzędzia do zarządzania licencjami. Tak zwany License Manager, dostępny w grupie programów Network Administration, przeznaczony jest dla administratorów instalacji komputerowych działających pod kontrolą Windows NT Server. Przy użyciu programu License Manager administratorzy mogą zdalnie zmieniać tryb licencji - z licencji typu „Per Server" na licencję typu „Per Seat" - dla programów i systemów takich, jak Windows NT Server 3.51 (udostępniający podstawowe usługi sieciowe zarządzania plikami, drukowania oraz komunikacji), SQL Server i SNA Server. Mogą też dodawać lub kasować licencje typu Client Access dla produktów z rodziny BackOffice. License Manager przyporządkowuje użytkowników systemu do odpowiednich licencji. Postanowienia licencyjne zapewniają legalny dostęp do oprogramowania serwera. W większości przypadków jeden użytkownik przypisany jest do jednego komputera. Kiedy jednak z danego, fizycznego komputera korzysta wielu użytkowników, administratorzy muszą podjąć dodatkowe kroki, na przykład tworząc lub modyfikując grupy licencyjne (w menu Options Advanced), aby zapewnić zgodność licencji.

Kiedy przy licencjonowaniu typu „na serwer" osiągnięty zostanie określony limit, Windows NT Server 3.51 nie zezwala na następne podłączenia się do danego serwera. Do tej chwili administrator systemu traktowany był jako jeden z równoległych użytkowników, których dotyczy limit licencji. W przypadku osiągnięcia tego limitu dostęp administratora jest oczywiście dopuszczony w celu odpowiedniego rozwiązania blokady.

Polityka upgrade

Spójna polityka upgrade produktów rodziny BackOffice bazuje na pojęciach Major Upgrade i Minor Upgrade.
Major Upgrade to zmiana oprogramowania, dla której zmienia się „część całkowita" numeru wersji programu (np. zmiana wersji SQL Server z 4.21a na 6.0). W przypadku takiej zmiany wersji użytkownik obciążany jest opłatą dotyczącą serwera oraz licencji dla stacji roboczych. Cena uaktualnienia wynosi 100 proc. ceny serwera za nową wersję (upgrade) serwera oraz 50 proc. ceny Client Access License za nowe wersje licencji dostępu dla stacji.
Minor Upgrade to zmiana oprogramowania, dla której zmienia się „część ułamkowa" numeru wersji programu. W przypadku takiej mniejszej zmiany wersji opłata obciąża serwer i wynosi 100 proc. pełnej ceny pakietu serwera. Są przypadki, w których uaktualnienie (upgrade) traktowane jest jako modyfikacja o mniejszej wadze niż Minor Upgrade. Na przykład za modyfikację typu „step-up", a nie upgrade, uważana jest zmiana systemu Windows NT Server z wersji 3.5 na wersję 3.51. Opłaty w takich niestandardowych przypadkach ustalane są oddzielnie dla danego przypadku. Przytoczony fragment tekstu umowy licencyjnej użytkownika końcowego na licen-

cję dostępu dla stacji roboczej użytkownika (Client Access License End User License Agreement) wyjaśnia zakres obowiązywania tej licencji i określa, kiedy wymagany jest upgrade tej licencji:

„Oprogramowanie Serwera opatrzone jest określonym numerem wersji (np. wersja „3.5"). Niniejsza licencja typu Client Access License upoważnia do dostępu lub wykorzystywania w inny sposób usług dowolnego Oprogramowania Serwera o tym samym (lub niższym), co wyszczególniony powyżej numerze wersji oprogramowania serwera na lewo od kropki dziesiętnej (jeśli więc na przykład wyszczególniony powyżej numer wersji wynosi „3.1", dozwolony jest dostęp lub wykorzystywanie Oprogramowania Serwera, opatrzonego numerem „3.5" lub „2.0", ale nie wersji „4.0")."

Umowa serwisowa typu Maintenance

Serwis typu Maintenance dostępny jest dla rodziny BackOffice produktów serwera w ramach umów licencyjnych Microsoft Select lub Microsoft Open License (MOLP). Serwis taki wykonywany jest na podstawie dwuletniej umowy z Microsoftem, w ramach której za stałą opłatą użytkownik otrzymywać będzie automatyczne modyfikacje typu update dla wszelkich nowych produktów - bez dodatkowej opłaty w przypadku programu Select i przy minimalnej opłacie w przypadku programu MOLP. Koszt roczny takiej umowy wynosi w przybliżeniu 15 proc. ceny ERP (Estimated Retail Price - oczekiwanej ceny detalicznej) pełnego pakietu serwera oraz licencji typu Client Access License. Na wszystkie produkty z rodziny BackOffice można podpisać taką umowę serwisową, niezależnie od numeru wersji oprogramowania. Innymi słowy oznacza to, że do podpisania umowy typu Maintenance niekoniecznie trzeba posiadać licencję na najnowszą wersję produktu. Ta właśnie cecha umowy typu Maintenance, pozwalająca na uzyskanie licencji na produkty na zasadzie „być na bieżąco i pozostać na bieżąco" przez okres 2 lat, stanowi o jej atrakcyjności w stosunku do jednorazowych modyfikacji typu upgrade. Mając zawartą umowę na serwis typu Maintenance użytkownik może dokładnie zaplanować swój budżet na oprogramowanie, opierając się na obniżonych cenach upgrade, niezmiennych przez cały okres obowiązywania umowy.

Uwagi dodatkowe:

- Jeśli użytkownik ma kilka serwerów w swym środowisku sieciowym, to w układzie „Per Server" dla każdego serwera musi wykupić co najmniej tyle związanych z nim licencji Client Access License, ile wynosi maksymalna liczba klientów podłączonych do danego serwera jednocześnie.
- Jednorazowa konwersja. W pewnym momencie użytkownik może dojść do sytuacji, w której liczba klientów podłączonych w danej chwili do serwera jest równa liczbie wszystkich stanowisk komputerowych wykorzystywanych jako stacje robocze. Wtedy opłaca się przejść na licencję w układzie „Per Seat". Istnieje możliwość dokonania takiej jednonorazowej konwersji z licencji „Per Server" na licencję „Per Seat" bez żadnej dodatkowej opłaty za zmianę typu licencji. Jeśli użytkownik wybierze to rozwiązanie, nie musi nawet powiadamiać o tym Microsoftu. Konwersja taka dokonywana jest tylko w jedną stronę - z licencji „na serwer" na licencje „na stanowisko" - nie jest natomiast możliwa konwersja w odwrotnym kierunku. Po dokonaniu takiej konwersji dla danego serwera, wszystkie komputery desktop, uzyskujące dostęp do tego serwera, muszą mieć licencje typu Client Access License.
- W hierarchii Systems Management Server wymagana jest co najmniej jedna licencja serwera dla oprogramowania SQL Server i jest ona nabywana oddzielnie. Konsola administratora systemu Systems Management Server wykorzystuje dane programu SQL Server, nie jest jednak wymagana licencja Client Access License dla programu SQL Server, niezależnie od tego, jaka jest liczba konsol administratorów systemu.
- Systems Management Server wykorzystuje w celu dystrybucji oprogramowania mechanizmy dzielonego dostępu do plików, udostępniane przez sieciowy system operacyjny. Jeśli jako serwer dystrybucyjny programu Systems Management Server działa pod Windows NT Server, wymagane jest posiadanie licencji Client Access Server dla systemu Windows NT Server. To samo dotyczy zresztą dowolnego innego sieciowego systemu operacyjnego wykorzystywanego jako serwer dystrybucyjny. Dowolny system, jaki dany użytkownik wykorzystuje do dostępu do plików i serwerów wydruku, powinien działać jako serwer dystrybucyjny w środowisku Systems Management Server.
- Licencja dla serwera wymagana jest dla każdego serwera korzystającego z kodu programu Systems Management Server (pierwotny, wtórny, pomocy itp.).
- Jeśli wymagane jest nabycie licencji serwera dla programu Systems Management Server, systemu Windows NT Server oraz SQL Server dla tego samego komputera, to tańsze jest wykupienie licencji BackOffice Server. *0/20/96*

SYSTEMY CZASU RZECZYWISTEGO

REDAGUJE ZESPÓŁ: dr inż. Zbigniew FRYŹLEWICZ (Politechnika Wrocławska), prof. dr hab. inż. Janusz GÓRSKI (Francusko-Polska Wyższa Szkoła Nowych Technik Informatyczno-Komunikacyjnych w Poznaniu), prof. dr hab. inż. Zbigniew HUZAR (Politechnika Wrocławska), prof. dr hab. inż. Tomasz SZMUC (Akademia Górniczo-Hutnicza), dr inż. Kazimierz WORWA (Wojskowa Akademia Techniczna), mgr inż. Zdzisław ŻURAKOWSKI (Instytut Automatyki Systemów Energetycznych we Wrocławiu) – przewodniczący zespołu.

ROK III ● LIPIEC ● 1996 ● 20

Metody projektowania oprogramowania

Metoda projektowania HOOD

Bogumiła Hnatkowska, Zbigniew Huzar
Centrum Informatyczne Politechniki Wrocławskiej

Wytwarzanie oprogramowania służącego do rozwiązania określonego problemu rozpoczyna się fazą analizy, która z jednej strony dostarcza usystematyzowanego opisu interesującego nas fragmentu rzeczywistości, a z drugiej strony jest podstawą sformułowania wymagań użytkownika budowanego systemu [5]. Zapis wymagań dotyczących tworzonego oprogramowania składa się na specyfikację, która stanowi podstawę do rozpoczęcia projektowania programów. Jedną z metod projektowania oprogramowania jest HOOD *(Hierarchical Object Oriented Design)*.

Projekt jest procesem podejmowania decyzji o sposobie przejścia od specyfikacji do implementacji za pomocą wybranych środków realizacyjnych. Projekt, jako dokument, pisany jest w języku, który powinien być zgodny z językiem specyfikacji, a także z przewidywanym językiem programowania. Podstawową czynnością projektowania jest określenie architektury oprogramowania [6], to znaczy:

- zdekomponowanie całego systemu na części składowe,
- określenie funkcji części składowych,
- określenie wzajemnych sprzężeń między częściami.

Implementacja polega na określeniu tego, jak działają poprzednio znalezione komponenty. Głównym dokumentem, który powstaje w tej fazie, jest tekst programu pisany w wybranym języku programowania. Specyfika implementacji systemów czasu rzeczywistego polega na tym, aby spełnić funkcjonalne i czasowe wymagania, nałożone na system przy wykorzystaniu ograniczonej ilości zasobów o ograniczonej wydajności.

HOOD [1, 3, 4] jest metodą projektowania oprogramowania ukierunkowaną na implementację w języku ADA. Głównym zadaniem metody jest systematyczne projektowanie architektury programów. Pierwsza wersja metody została opracowana w 1987 r. w wyniku kooperacji francuskich firm CISI Ingenierie i Matra Espace z duńską firmą CRI A/S, na zlecenie Europejskiej Agencji Kosmicznej ESA *(European Space Agency)*. Metoda HOOD była stosowana w wielu dużych przedsięwzięciach programistycznych, m.in. w zastosowaniach militarnych i energetyce jądrowej. W miarę upływu czasu metodę modyfikowano (formalizowano) w celu umożliwienia budowy wspomagającego oprogramowania narzędziowego. Metoda ciągle rozwija się, a zasadniczym motywem obecnego rozwoju jest uwzględnienie metod analizy obiektowej i programowania obiektowego.

Punktem wyjścia do zastosowania metody HOOD jest specyfikacja systemu. Metoda nie narzuca żadnego języka ani sposobu analizy specyfikacji, co z jednej strony jest zaletą, a z drugiej wadą, gdyż nie daje wskazówek systematycznego wykorzystania informacji nagromadzonych w czasie analizy. Domyślnie można zakładać, że system jest opisany w konwencji obiektowej, a specyfikacja jest napisana w języku naturalnym. Końcowym wynikiem zastosowania metody HOOD jest szczegółowy projekt, który może być wykorzystany do automatycznej generacji programu w docelowym języku programowania.

Elementy podstawowe

Metoda HOOD wprowadza własną notację graficzną do przedstawiania obiektów i zachodzących między nimi związków (rys. 2, 3, 4). Dodatkowo metoda wprowadza notację tekstową, w pewnym zakresie równoważną notacji graficznej, która przeznaczona jest przede wszystkim do przedstawiania szczegółów implementacyjnych.

Obiekty

Obiekt jest podstawowym, jednoznacznie odróżnialnym elementem składowym modelu budowanego systemu. Jego określenie jest wyraźnie ukierunkowane na język

ADA – obiekt jest enkapsulacją (hermetyzacją) danych, operacji dostępu i zmiany tych danych oraz definicji odpowiednich typów danych, stałych i wyjątków. Odpowiednikami obiektu w sensie metody HOOD są w języku ADA pakiety i zadania.

Operacje, definicje typów danych, stałych oraz wyjątków stanowią *interfejs* obiektu. Obiekt współdziała z otoczeniem za pośrednictwem interfejsu, który określa usługi oferowane przez dany obiekt i wskazuje usługi innych obiektów, z których dany obiekt korzysta. Współdziałanie może polegać na wywoływaniu przez otoczenie obiektu jego operacji, albo też na wywoływaniu przez dany obiekt operacji innych obiektów należących do jego otoczenia. W obu przypadkach mówi się, że dany obiekt wykonuje *interakcje* z innymi obiektami. Istotą interakcji jest komunikacja, czyli przesyłanie danych między obiektami. Przez daną rozumie się wartość zmiennej wewnętrznej obiektu. Zestaw wartości zmiennych określa *stan* obiektu. Stan obiektu może ulec zmianie tylko w wyniku interakcji obiektu z innym obiektem, albo w wyniku wykonania przez obiekt pewnej *akcji wewnętrznej*, niezależnej od jego otoczenia.

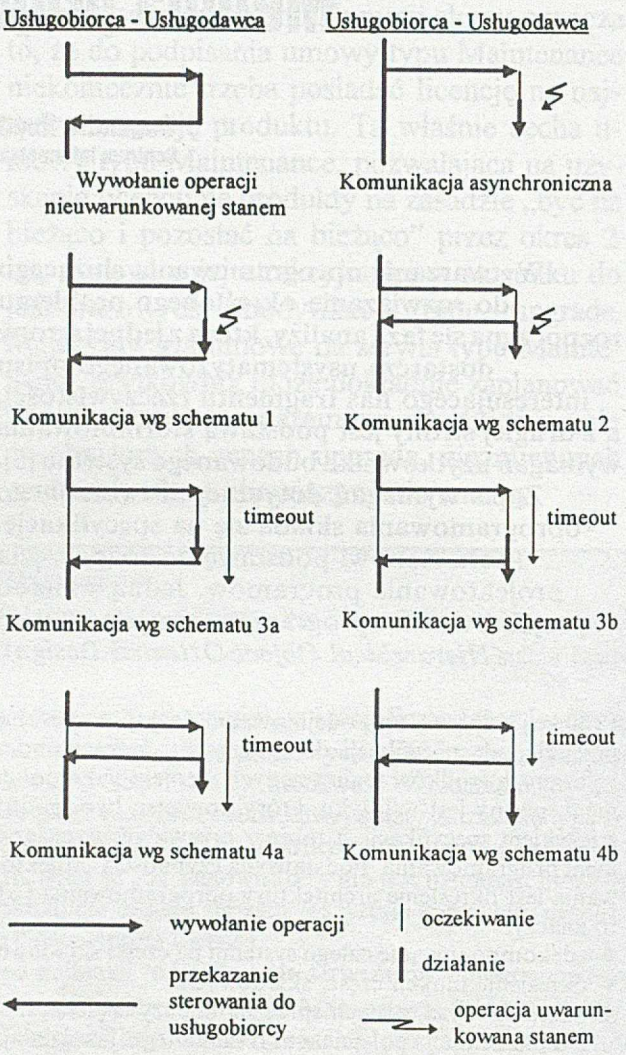
Obiekt udostępnia swoje usługi od chwili utworzenia aż do chwili skasowania. W ramach usług obiekt może oferować dostęp do definicji typów danych, odczytanie wartości stałych oraz wywołanie operacji. Efektem wywołania operacji z odpowiednimi parametrami wejściowymi może być przekazanie wartości albo *wyjątków*. Wyjątki należy rozumieć jako sygnalizację sytuacji, które uniemożliwiają wykonanie operacji bądź wskazują na powstanie błędów podczas wykonywania operacji. Obiekt jest charakteryzowany przez własności statyczne (interfejs z otoczeniem) i dynamiczne (zachowanie obiektu). Własności dynamiczne obiektu określają dopuszczalne ciągi interakcji danego obiektu z jego otoczeniem.

Obiekt składa się z *części widzialnej (publicznej)* i *części wewnętrznej (ukrytej)*. Część widzialna definiuje interfejs obiektu, zaś jego część ukryta stanowi implementację elementów z części widzialnej. W jej skład wchodzą definicje typów danych, stałych, operacji, wyjątków, zmiennych oraz obiektów wewnętrznych. Użyte tu pojęcia są rozumiane tak, jak w językach programowania, takich jak ADA czy C++. Obok pojęcia obiektu, metoda HOOD posługuje się pojęciem *klasy obiektów*, rozumianej jako wzorzec (być może parametryzowany), na podstawie którego można generować egzemplarze konkretnych obiektów. Odpowiednikiem klasy obiektów jest typ zadaniowy lub pakiet generyczny ADY.

Operacje

W okresie swego istnienia obiekt może wykonywać pewne *aktywności* (akcje wewnętrzne i interakcje) i zmieniać swój stan. Podstawowa aktywność obiektu wynika z wykonywania operacji wywołanych przez inne obiekty, czyli z jego roli usługodawcy. Klasyfikacja schematów współpracy obiektów: *usługodawcy* z *usługobiorcą* (rys. 1) bierze pod uwagę, czy obiekty te wykonują się *sekwencyjnie* czy *współbieżnie*. Gdy obiekty wykonują się względem siebie sekwencyjnie, to zachodzi najprostszy schemat współpracy, w którym operacja wywoływana przez obiekt-usługobiorcę jest natychmiast wykonywana przez obiekt-usługodawcę, a po jej zakończeniu wyniki są natychmiast zwracane do usługobiorcy, który zawiesza swoją aktywność na okres wykonywania operacji. Te, które są wykonywane zgodnie z tym schematem, określa się jako *operacje nieuwarunkowane stanem usługodawcy*. Na podstawie takiego schematu odbywa się w języku ADA wywoływanie podprogramów pakietu przez inny pakiet.

W przypadku, gdy współpracujące obiekty są względem siebie współbieżne, są możliwe schematy współpracy *asynchronicznej* i *synchronicznej*. Operacje wykonywane zgodnie z tymi schematami noszą nazwę *operacji uwarunkowanych stanem*. Wyraża to fakt, że wywołanie operacji nie zawsze musi być przyjęte przez usługodawcę. Wynikiem wywołania operacji może być wykonanie oczekiwanego obliczenia albo odmowa wykonania obliczenia, sygnalizowana przez przekazanie wyjątku. To, która z tych sytuacji zajdzie, jest uwarunkowane stanem wewnętrznym obiektu-usługodawcy.



Rys. 1. Schematy współpracy obiektów

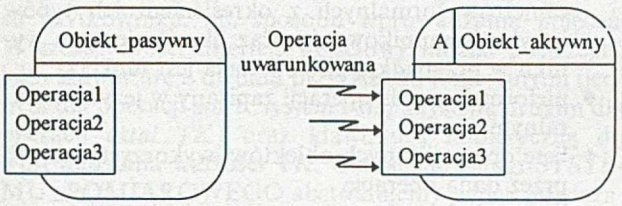
Współpraca asynchroniczna zakłada, że pomiędzy współpracującymi obiektami istnieje system przekazywania wiadomości, do którego obiekt-usługobiorca może w dowolnym momencie wysłać komunikat przeznaczony do obiektu-usługodawcy. Usługobiorca po wysłaniu komunikatu kontynuuje swoje zadanie. W języku ADA na podstawie takiego schematu może odbywać się wywoływanie wejść zadań.

Współpraca synchroniczna polega na tym, że obiekty ustalają między sobą *stałe połączenie* (spotkanie) na okres wymiany informacji związanej z wykonaniem operacji.

W ramach współpracy synchronicznej są możliwe cztery szczegółowe schematy współpracy:

1. Obiekt-usługobiorca wywołuje operację i czeka tak długo, aż obiekt-usługodawca zacznie wykonywać operację i przekaże obliczone wyniki.
2. Obiekt-usługobiorca wywołuje operację i czeka tak długo, aż obiekt-usługodawca potwierdzi wywołanie operacji.
3. Obiekt-usługobiorca wywołuje operację i czeka na rozpoczęcie wykonywania operacji i przekazania obliczonych wyników przez zadany okres przeterminowania (a). Jeżeli wyniki nie zostaną dostarczone przed upływem okresu przeterminowania, usługobiorca przerywa połączenie i wznawia swą aktywność (b).
4. Obiekt-usługobiorca wywołuje operację i czeka na potwierdzenie wywołania przez zadany okres przeterminowania (a). Jeżeli potwierdzenie nie zostanie dostarczone przed upływem okresu przeterminowania, usługobiorca przerywa połączenie i wznawia swą aktywność (b).

Wyróżnia się dwa rodzaje obiektów: *aktywne i pasywne* (rys. 2). Obiekty pasywne oferują wyłącznie operacje nieuwarunkowane, a aktywne oferują przynajmniej jedną operację uwarunkowaną.



Rys. 2. Oznaczenie obiektu pasywnego i aktywnego

Proces projektowania

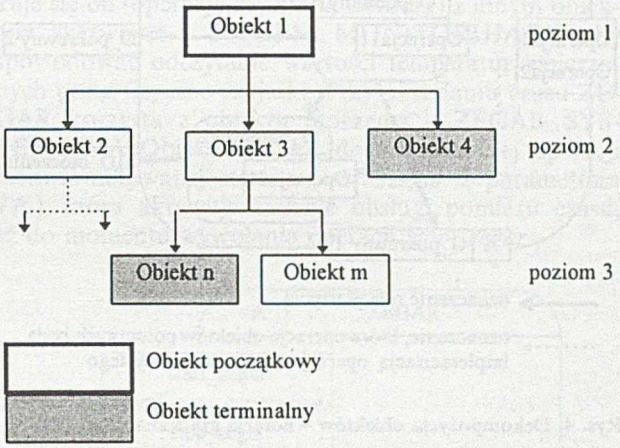
Metoda HOOD jest zgodna z kaskadowym modelem rozwoju oprogramowania. Projektowanie jest procesem zstępującym, a projektowany system jest początkowo widziany jako „czarna skrzynka" reprezentowana przez jeden obiekt początkowy. Jest on dekomponowany na dalsze obiekty, które również mogą podlegać dekompozycji. W ten sposób powstaje drzewo projektowe HDT (*HOOD Design Tree* – rys. 3). Kolejne poziomy drzewa tworzą hierarchię poziomów abstrakcji, w której niższe poziomy są uszczegółowieniem poziomów wyższych.

Metoda wyróżnia trzy aspekty projektowania:

- funkcje budowanego systemu,
- rozproszenie (przestrzenne rozmieszczenie) jego komponentów,
- fizyczne urządzenia wykonawcze.

Zasadniczy aspekt dotyczy *projektowania funkcjonalnego*, które polega na określeniu struktury systemu, tzn. ustaleniu elementów składowych i określeniu związków między nimi. Podstawowe zasady, na których opiera się projektowanie na danym poziomie abstrakcji, dotyczą:

1. Użycia *obiektów* jako podstawowych jednostek dekompozycji (modularyzacji). Obiekt funkcjonuje w otoczeniu, któremu udostępnia swoje usługi za pośrednictwem ustalonego interfejsu i z usług którego korzysta. Obiekt jest tu rozumiany jako jednostka modularyzacji oprogramowania i nie powinien być utożsamiany bezpośrednio z obiektem stanowiącym komponent rzeczywistego systemu, którego obraz jest tworzony w fazie analizy. W kontekście języka ADA obiekty mogą być rozumiane jako podprogramy, pakiety lub zadania.
2. Określenia *związku* łączącego obiekty. Podstawowym związkiem jest *związek użycia*, który zachodzi między obiektem-usługobiorcą a obiektem-usługodawcą i oznacza, że usługobiorca korzysta z usług oferowanych przez usługodawcę. Związek użycia powinien tworzyć relację hierarchiczną, a dokładniej acykliczny, skierowany graf, którego wierzchołkami są obiekty, a łuki prowadzą od obiektów-usługobiorców do obiektów-usługodawców.
3. Określenia *zasad współpracy* obiektów. Współpraca opiera się na wymianie informacji przez przesyłanie komunikatów. Przesyłaniu komunikatów towarzyszą różne modele synchronizacji, w szczególności różne sposoby reagowania na upływ czasu.



Rys. 3. Przykład drzewa projektowego HOOD

Zasady przechodzenia między kolejnymi poziomami abstrakcji, czyli implementacja obiektów danego poziomu za pośrednictwem obiektów niższego poziomu, są omówione dalej. Ogólnie, implementacja wiąże się z określeniem związku *zawierania* (zagnieżdżania obiektów): obiekt potomny może stanowić element składowy obiektu macierzystego. Związek ten powinien wyznaczać strukturę drzewa. Drzewem ma być graf skierowany, którego wierzchołki są obiektami, a łuki prowadzą od obiektu macierzystego do jego obiektów potomnych (rys. 3). Pozostałe aspekty projektowania dotyczą projektowania programu rozproszonego. W kontekście całej metody mają one znaczenie drugoplanowe.

Zasady dekompozycji obiektów

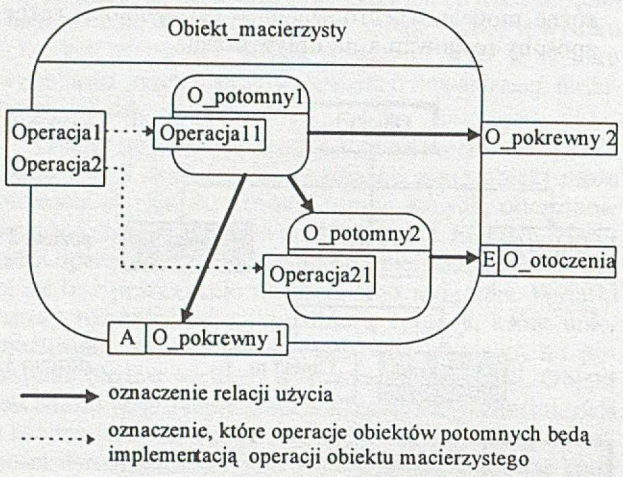
Istotą dekompozycji jest reprezentacja (implementacja) operacji obiektu macierzystego przez obiekty potomne. Każda operacja obiektu macierzystego może być implementowana przez:

- operację obiektu potomnego, co oznacza, że w trakcie wykonywania operacji obiektu macierzystego następuje wywołanie operacji obiektu potomnego;

- przez kilka obiektów potomnych; w tym przypadku wprowadza się wydzielony obiekt potomny, zwany obiektem sterującym, który odwzorowuje operacje obiektu macierzystego na operacje obiektów potomnych.

Obiekt, którego operacje nie są dalej dekomponowane, określa się mianem obiektu *końcowego (terminalnego)*. Implementację operacji takiego obiektu wyraża się w pseudokodzie lub w wybranym języku programowania. Obiekty potomne implementujące operacje obiektu macierzystego mogą wykorzystywać usługi innych obiektów, nie zagnieżdżonych w tym samym obiekcie macierzystym. Mogą to być (rys.4):

- obiekty *pokrewne* (używane przez dany obiekt macierzysty);
- obiekty *otoczenia* (zewnętrzne w stosunku do budowanego oprogramowania).



Rys. 4. Dekompozycja obiektów – notacja graficzna

Metoda wprowadza sformalizowany *schemat opisu obiektu* (ang. *Object Description Skeleton*). Jest to formalny dokument, który zawiera następujące informacje:

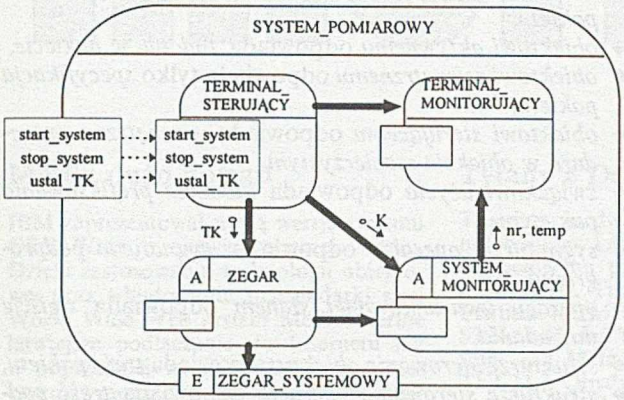
1. Opis ogólny obiektu, na który składa się:
   - określenie nazwy i rodzaju obiektu (obiekt aktywny lub pasywny, wewnętrzny lub zewnętrzny, sterujący lub niesterujący, węzeł wirtualny itp.),
   - nieformalny, słowny opis roli i miejsca obiektu w projektowanym systemie,
   - opis ograniczeń implementacyjnych lub synchronizacyjnych, na przykład określenie sprzężeń ze sprzętem przez przerwania, sposób współpracy z zegarami systemowymi, sposób zapobiegania blokadom itp.
2. Interfejs obiektu, przedstawiający w konwencji języka ADA formalny opis zasobów (typy, stałe, operacje oraz wyjątki) udostępnianych przez obiekt.
3. Struktura sterowania obiektu aktywnego, na którą składa się:
   - opis interfejsu, który zawiera nieformalny opis operacji uwarunkowanych obiektu oraz sformalizowaną listę tych operacji,
   - opis interakcji pomiędzy obiektem a użytkownikami wywołującymi operacje obiektu. Dla obiektów terminalnych opis jest wyrażony najpierw w pseudokodzie, a następnie w języku docelowym. W przypadku obiektów dekomponowanych na obiekty potomne, ta część, zamiast fragmentów pisanych w pseudokodzie i kodzie, zawiera listę aktywnych obiektów potomnych, które implementują operacje obiektu macierzystego.
4. Zestawienie usług innych obiektów, używanych przez dany obiekt.
5. Przepływy danych i wyjątków, określające nazwy danych i wyjątków przesyłanych między danym obiektem a obiektami współpracującymi.
6. Opis wewnętrznych składowych obiektu (odpowiednik opisu treści obiektu), tj.:
   - listę obiektów potomnych danego obiektu,
   - listę typów danych związanych z danym obiektem,
   - listę stałych związanych z danym obiektem,
   - listę operacji obiektu macierzystego ze wskazaniem na obiekty potomne, które je implementują, oraz – w przypadku obiektów końcowych – listę z opisem operacji wewnętrznych obiektu,
   - listę wyjątków związanych z danym obiektem,
   - listę deklaracji zmiennych obiektu.
7. Szczegółowy opis operacji obiektów końcowych, który powinien prowadzić do zapisu kodu w języku ADA. Opis pojedynczej operacji zawiera:
   - deklarację operacji, tzn. nazwę operacji, listę parametrów formalnych z określeniem ich typów i trybu komunikowania oraz określenie typu wyniku w przypadku, gdy operacja jest funkcją,
   - nieformalny opis operacji zapisany w języku naturalnym,
   - listę operacji innych obiektów, wykorzystywanych przez daną operację,
   - listę wyjątków generowanych przez daną operację,
   - nieformalny, wyrażony w języku naturalnym, opis obsługi wyjątków w obrębie danej operacji,
   - opis operacji zapisany w pseudokodzie; opis ten powinien być uściśleniem wcześniejszego opisu nieformalnego,
   - opis operacji w języku ADA, jako kolejne uściślenie poprzedniego opisu w pseudokodzie (pojęcie pseudokodu jest rozumiane tu zupełnie nieformalnie, jako notacja wprowadzająca pewne skróty notacyjne do opisu w języku naturalnym),
   - opis końcowej instrukcji operacji (instrukcji definiującej wynik jej obliczeń i określenie warunków zakończenia operacji).
8. Sposób kończenia pracy obiektu, przez wskazanie innego obiektu i operacji, która powoduje zakończenie danego obiektu.

Przykład projektowania funkcjonalnego

W celu ilustracji metody – dokładniej dekompozycji obiektu – rozpatrzmy system monitorowania temperatury złożony z K czujników mierzących temperaturę. System cyklicznie, co TK minut, odczytuje wartość temperatury i wyświetla ją na monitorze. W przypadku braku odpowiedzi od czujnika w ciągu 1 s, system wyświetla informacje o uszkodzeniu. Użytkownik może w dowolnym momencie włączyć lub wyłączyć system pomiarowy za pomocą panelu sterowania. Operator systemu może też ustalić nowy okres TK cyklicznego odczytu czujników.

Przedstawiony nieformalny opis potraktujmy jako specyfikację systemu monitorowania temperatury. Pro-

jektowanie systemu rozpoczniemy od określenia interfejsu obiektu bazowego – SYSTEM_POMIAROWY – oraz jego dekompozycji na obiekty potomne (rys. 5).

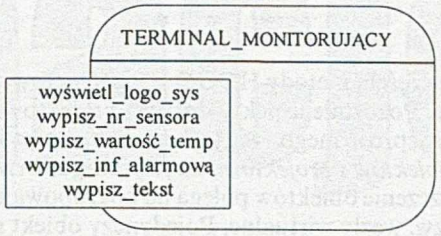


Rys. 5. Dekompozycja SYSTEMU_POMIAROWEGO na obiekty potomne

Operacje interfejsu bazowego obiektu SYSTEM_ _POMIAROWY wynikają bezpośrednio ze specyfikacji. Projekt nie uwzględnia szczegółów rzeczywistego interfejsu użytkownika, np. sposobu wprowadzania poleceń. W szczególności interfejs wejściowy można postrzegać jako klawiaturę z dwoma przyciskami funkcyjnymi (jednym dla operacji *start_system* i *stop_system*, drugim dla operacji *ustal_TK*) oraz klawiaturą numeryczną do wprowadzania wartości TK. Określając usługi SYSTEMU_POMIAROWEGO abstrahujemy od ich powiązania z obsługą urządzeń wejścia. Podjęto decyzję, że SYSTEM_POMIAROWY będzie obiektem pasywnym, tzn. nie będzie miał operacji uwarunkowanych stanem. W praktyce oznacza to, że obiekt będzie reprezentowany przez pakiet ADY, a jego operacje będą sekwencyjnymi podprogramami, których wykonanie powoła do życia obiekty aktywne (zadania ADY).

Obiekt SYSTEM_POMIAROWY zdekomponowano na cztery obiekty potomne. Podany niżej opis można traktować jako ogólny opisu obiektu (punkt 1 w schemacie opisu obiektu). Obiekt TERMINAL_STERUJĄCY odpowiada za obsługę wejścia użytkownika (wymusza zmiany stanu obiektu macierzystego). Jest to obiekt terminalny, który nie podlega dalszej dekompozycji. Przy realizacji usług TERMINAL_STERUJĄCY korzysta z obiektów potomnych (relacja użycia). Nieformalna specyfikacja jego usług wygląda następująco:

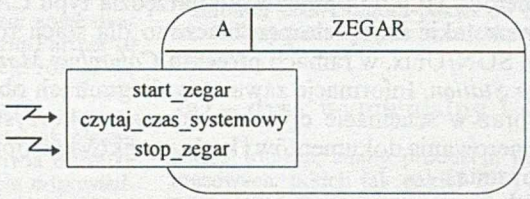
- *start_system* (*K* – liczba sensorów, *TK* – początkowy okres próbkowania): wyświetla logo systemu na TERMINALU_MONITORUJĄCYM, inicjuje pracę SYSTEMU_MONITORUJĄCEGO fizyczne czujniki, inicjuje pracę ZEGARA wymuszającego odczyt temperatury co *TK* minut;
- *stop_system:* kończy pracę ZEGARA, SYSTEMU_MONITORUJĄCEGO oraz wypisuje napis „KONIEC PRACY" na TERMINALU_MONITORUJĄCYM;
- *ustal_TK* (*TK* – nowy okres próbkowania): kończy pracę ZEGARA i inicjuje nowy ZEGAR z nowym okresem próbkowania *TK*; jeśli $K*(1s + czas\_odczytu\_sensora) > TK$ (system nie zdąży odczytać czujników), wypisuje napis „ZBYT MAŁA WARTOŚĆ *TK*" na TERMINALU_MONITORUJĄCYM (dotychczasowa wartość *TK* bez zmian).



Rys. 6. Usługi obiektu TERMINAL_MONITORUJĄCY

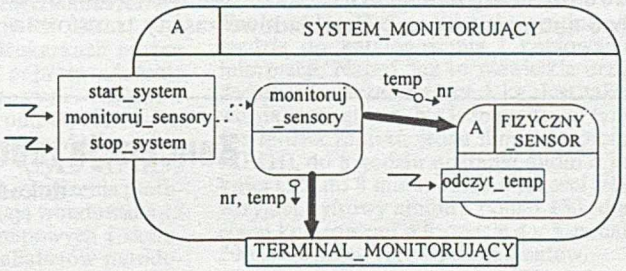
Obiekt TERMINAL_MONITORUJĄCY (rys. 6) odpowiada za obsługę wyjścia użytkownika. Jest to obiekt pasywny, dostarczający usług zarówno obiektowi TERMINAL_STERUJĄCY *(wyświetl_logo_sys, wypisz_tekst)*, jak i obiektowi SYSTEM_MONITORUJĄCY *(wypisz_nr_sensora, wypisz_wartość_temp)*. Pełne deklaracje operacji TERMINALA_MONITORUJĄCEGO wymagają oczywiście uszczegółowienia.

Aktywny obiekt ZEGAR (rys. 7) odpowiada za pomiar względnego czasu w systemie. Co *TK* minut komunikuje się on (operacja *monitoruj_sensory*) z innym obiektem aktywnym – SYSTEM_MONITORUJĄCY, aby spowodować odczytanie wartości temperatur zmierzonych przez fizyczne czujniki. Przy określaniu czasu ZEGAR korzysta z obiektu otoczenia – ZEGAR_SYSTEMOWY. Obiekt ZEGAR dostarcza jednej operacji nieuwarunkowanej stanem (*start_zegar* z parametrem *TK*), która aktywuje zadanie obsługi pomiaru czasu, aż do momentu wywołania operacji *stop_zegar*.



Rys. 7. Usługi obiektu ZEGAR

Aktywny obiekt SYSTEM_MONITORUJĄCY (rys. 8) odpowiada za odczytanie wartości temperatury.



Rys. 8. Dekompozycja obiektu SYSTEM_MONITORUJĄCY

Do komunikacji z fizycznymi urządzeniami wydzielono osobny obiekt potomny – FIZYCZNY_SENSOR, przy czym komunikacja z tym obiektem zachodzi zgodnie ze schematem komunikacji z przeterminowaniem. Gdy operacja *monitoruj_sensory* nie otrzyma wyników pomiarów w ciągu 1 s, na spokrewnionym obiekcie TERMINAL_MONITORUJĄCY zostanie wyświetlona informacja o uszkodzeniu czujnika. W przeciwnym przypadku na TERMINALU_MONITORUJĄCYM są wyświetlane wartości odczytane z sensorów.

* * *

Główną częścią metody HOOD jest projektowanie funkcjonalne. Pozostałe aspekty dotyczą projektowania programu rozproszonego, w tym *przestrzennego rozmieszczenia obiektów* i *projektowania fizycznego*. Przestrzenne rozmieszczenie obiektów polega na pogrupowaniu obiektów w tzw. węzły wirtualne. Pojedynczy obiekt może być powielony i przydzielony do kilku węzłów wirtualnych. Pomiędzy nimi mogą zachodzić – tak jak pomiędzy obiektami – związki usługowe i związki zawierania. Projektowanie fizyczne polega na przypisaniu węzłom wirtualnym fizycznych urządzeń (procesorów i pamięci), które będą stanowić ich środowisko wykonawcze. Pojedynczy węzeł fizyczny może być środowiskiem wykonawczym dla wielu węzłów wirtualnych. Końcowy węzeł wirtualny, który nie zawiera w sobie innych węzłów wirtualnych, musi być wykonywany na jednym procesorze.

Język ADA83 nie pozwalał implementować programów rozproszonych. Taka możliwość pojawia się w języku ADA95 [2] i polega na odpowiednim użyciu tzw. *pragm*, czyli dyrektyw kompilatora, którymi uzupełnia się tekst programu. Pragmy nie mają wpływu na znaczenie programu, mogą tylko określać sposób jego wykonania. Wykorzystuje się je do wskazania węzłów wirtualnych i określenia sposobów komunikacji między nimi. Dopuszcza się m.in. komunikację za pośrednictwem wspólnych danych oraz zdalne wywołania podprogramów. Dla wsparcia projektowania oprogramowania metodą HOOD opracowano narzędzia typu CASE. Pierwsze takie narzędzie przeznaczono dla stacji roboczych SUN/Unix, w ramach projektu *Columbus Manned Space Station*. Informacje zawarte w diagramach obiektów oraz w schemacie opisu obiektu są wykorzystane do generowania dokumentów i kodu wynikowego oprogramowania.

Generowanie kodu wynikowego należy rozumieć z pewnym zastrzeżeniem. Generacja składa się z dwóch faz. Faza pierwsza jest odwzorowaniem struktury projektu (zbiór obiektów i związków pomiędzy nimi) na strukturę programu (zbiór jednostek programowych i związków pomiędzy nimi). Faza ta jest określona w pełni i może być automatyzowana. Przykładowe zasady transformacji, które stosuje się do wyznaczenia struktury programu na podstawie projektu, są następujące:

- *nazwie obiektu* odpowiada *nazwa pakietu*,
- *obiektowi pasywnemu* odpowiada *specyfikacja i treść pakietu*,
- *obiektowi aktywnemu* odpowiada *zadanie w pakiecie*,
- *obiektowi zewnętrznemu* odpowiada tylko *specyfikacja pakietu*,
- *obiektowi sterującemu* odpowiada *wewnętrzna procedura w obiekcie macierzystym*,
- *związkowi użycia* odpowiada *klauzula prefiksowania pakietem*,
- *sygnaturze operacji* odpowiada *sygnatura podprogramu*,
- *operacji uwarunkowanej stanem* odpowiada *wejście do zadania*,
- *strukturze sterowania obiektu* odpowiada *treść pakietu*,
- *strukturze sterowania operacją* odpowiada *treść podprogramu*,
- *klasie* odpowiada *pakiet generyczny*,
- wielu innym elementom, np. *przepływom danych i wyjątków*, odpowiadają *komentarze* w języku ADA, które wymagają „ręcznego" przetransformowania na kod wynikowy.

Druga faza polega na wypełnieniu treścią jednostek programowych na podstawie informacji zawartych w szkieletach opisów odpowiadających im obiektów. Realizacja tej fazy zależy od stopnia szczegółowości opisu operacji obiektu w języku ADA (punkt 7 w szkielecie opisu obiektu), i dlatego nie zawsze może prowadzić do pełnego kodu programu.

LITERATURA

[1] Delatte B., Heitz M., Muller J.F.: HOOD – Reference Manual 3.1, Masson, Paris, 1993

[2] ISO/IEC 8652:1996 (E), Information technology – Programming languages – Ada. Ada Reference Manual, Version 6, December 1994

[3] Nielsen K., Shumate K.: Designing Large Real-Time Systems with ADA, Multiscience Press Inc., 1988

[4] Robinson P.J.: Hierarchical Object-Oriented Design, Prentice Hall, 1992

[5] Solvberg A., Kung D. C.: Information Systems Engineering, Springer-Verlag, 1993

[6] Turner K. J.: Specification Architecture in a Communication Context, Internal Report, Dept. of Coomputing Science and Mathematics, University of Stirling, March 1995.

Koncepcja modelu dla obiektowej...

dokończenie ze s. 15

LITERATURA

[1] Abiteboul S., Beeri C.: On the Power of Languages for the Manipulation of Complex Objects. „International Workshop on Theory and Applications of Nested Relations and Complex Objects", 1987

[2] Andrews T., Harris C., Duhl J.: The Ontos Object Database (raport techniczny). Ontologic Inc. Burlington, MA, 01803, 1990

[3] Codd E.: A Relational Data Model for Large Shared Data Banks, „Communications of AEM", 13(6), lipiec 1970

[4] Copeland G., Maier D.: Making Smalltalk a Database System. „ACM SIGMOD International Conference", lipiec 1984

[5] Delobel C., Lecluse C., Richard P.: Databases: From Relational to Object-Oriented Systems. International Thomson Publishing, 1995

[6] Eckert G., Golder P.: Improving object-oriented analysis. „Information and Software Technology", 36(2), s. 67–86, 1994

[7] Kuper G.: The Logical Data Model: a New Approach to Database Logic (praca doktorska), Stanford University, wrzesień 1985

[8] Lipski W.: On Databases with Incomplete Information. „Journal of the ACM", 28(1), 1981

[9] Stonebraker M.: Object-Relational Database Systems (raport techniczny), Illustra Information Technologies, Inc.

[10] Stonebraker M., Rowe L., Hirohama: The Implementation of POSTGRES. „IEEE Transaction on Knowledge and Data Engineering", 2(1), marzec 1990

[11] Tsichritzis D.C., Lochovsky F.H.: Modele danych. Warszawa, Wydawnictwa Naukowo-Techniczne, 1990

[12] Wegbreit P.: ACM Conference on Data: Abstraction Definition and Structure. ACM, 1977

[13] Wieczerzycki W.: Obiektowe bazy danych. „Informatyka", 1993(1), s.: 8–11.

informacje

Merlin dużo potrafi

IBM zaprezentował nową wersję systemu OS/2 Warp o nazwie roboczej Merlin. Dzięki zastosowaniu technologii obiektowej oraz wbudowaniu przeglądarki sieci World Wide Web system Merlin oferuje łatwiejsze podłączanie do Internetu niż w poprzedniej wersji OS/2 Warp. Użytkownicy mogą umieścić na ekranie ikony reprezentujące najczęściej odwiedzane strony WWW i klikając je, natychmiast uruchomić witrynę. Przeszukiwacze Zasobów (ang. *Uniwersal Resource Locators*), odpowiadające każdej z ikon, mogą być ładowane z i na dyskietki oraz grupowane w foldery. Merlin umożliwi m.in. łatwą wymianę zbiorów z serwerami FTP (ang. *File Transfer Protocol*) a także podłączenie komputerów osobistych do systemów mainfraime. Jest ono realizowane na zasadzie tunelowania protokołu wykorzystywanego w IBM Personal Communications/3270 w protokole TCP/IP. Użytkownicy Merlina będą mogli szybko przeglądać i pozyskiwać zasoby sieciowe dzięki ich udostępnieniu w okienku sieciowym, w postaci ikon. Użytkownik może przenieść na przykład arkusz Lotusa 1-2-3 z serwera NetWare na serwer Warp Server. Merlin może też pracować jako stacja/serwer w sieciach typu peer-to-peer (bezserwerowych). Będzie on dostarczany z pojedynczym requesterem (programem stacji roboczej przekazującym żądania ze stacji do serwera), obsługującym system OS/2 Warp Server i inne zgodne z nim systemy sieciowe. Jest on połączeniem requestera sieci lokalnej LAN i requestera sieci peer-to-peer, które w poprzednich wersjach były oddzielne. Merlin ma być dostępny w drugiej połowie bieżącego roku.

Obiektowe środowisko programowania Autodesk

Po trwających pięć miesięcy testach wersji beta, firma Autodesk rozpoczęła bezpłatne dostarczanie pakietu AutoCAD Runtime Extension (ARX) Software Developers Kit (SDK) dla ponad 3,5 tys. zarejestrowanych programistów z całego świata. ARX to obiektowe środowisko programowania firmy Autodesk, pozwalające na tworzenie szybkich, obiektowych aplikacji z zakresu projektowania i kreślenia, oraz wykorzystujące możliwości AutoCADAa Release 13, którego premiera odbyła się w listopadzie 1994 r. Zapewniając ścisłą integrację z jądrem AutoCADa, ARX SDK po raz pierwszy umożliwia niezależnym producentom tworzenie obiektowych aplikacji, szybkich i łatwych w obsłudze. Oprócz AutoCADa do aplikacji Autodesku opartych na ARX zaliczają się: The Mechanical Desktop, ADE 2.0 i AutoCAD MAP.

Tytany z Della

Titanium to dwa nowe modele szybkich notebooków Latitude XPi Della z niskonapięciowymi procesorami Pentium 100 i 133 MHz. Titanium P100 SD zawiera Pentium 100 MHz z wyświetlaczem SVGA STN 10,4", zaś Titanium P133 ST – Pentium 133 MHz z wyświetlaczem SVGA TFT 11,3. Obydwa modele wyposażone są w 8 MB EDO RAM, rozszerzalne do 40 MB, 128-bitowy akcelerator graficzny NeoMagic NM2070 (odtwarza pliki wideo w standardzie MPEG), zewnętrzną pamięć podręczną typu cache L2 256 kB, wyjmowany dysk twardy o pojemności 540 do 1,2 GB, architekturę PCI, zasilanie z baterii litowo-jonowych, 16-bitową kartę dźwiękową zgodną z Sound Blaster Pro, wbudowany głośnik i mikrofon, optyczny trackball, pracujące na podczerwieni łącze transmisji danych w standardzie IrDA-1.

Drukarki laserowe od 600 do 1200 pkt/cal

Na naszym rynku pojawiły się nowe drukarki laserowe FUJITSU – PrintPartner 10 oraz PrintPartner 14. Modele różnią się tylko nieznacznie szybkością wydruku oraz wyglądem. Obie drukarki są wyposażone w procesor SPARC Lite taktowany zegarem 20 MHz, co umożliwia obróbkę danych i wydruk z szybkością odpowiednio 10 i 14 str/min z rozdzielczością 6000 × × 600 pkt/cal. Zastosowanie specjalnego mechanizmu FEIT (ang. *Fujitsu Enhanced Imaging Technology*) umożliwia uzyskanie wydruku z rozdzielczością 1200 × 1200 pkt/cal. Zastosowane techniki zarządzania pamięcią pozwalają na wydrukowanie strony A4 z grafiką przy tylko 2MB standardowej pamięci RAM dla modelu PP 1000 i 3 MB dla PP 14. Pamięć w drukarkach można rozszerzyć do 35 MB poprzez włożenie standardowych 32-bitowych układów SIMM. Urządzenia są kompatybilne z HP Laser 4 emulując PCL 5e. Możliwe jest wysyłanie rozkazów w języku HPGL/2 i używanie drukarki jako urządzenia plotującego. Obie drukarki mają wbudowane 45 rodzajów czcionek bitmapowych i skalowalnych, a także wiele alfabetów narodowych, w tym kilka standardów polskich liter. Można wyposażyć je w dodatkowy osprzęt w celu dopasowania do indywidualnych potrzeb użytkownika.

Zarządzanie z PRO/MIS-em

System Zarządzania PRO/MIS autorstwa CSBI otrzymał nagrodę podczas ogólnopolskiego sympozjum „Systemy informatyczne w energetyce" (Bielsko Biała, kwiecień br.). Nowym elementem obecnej wersji PRO/MIS jest zestaw modułów umożliwiających prowadzenie rachunkowości zarządzającej. System nie tylko rejestruje zaistniałe zdarzenia gospodarcze, ale pozwala również na bieżące monitorowanie działalności przedsiębiorstwa, konfrontację osiąganych wyników z planowanymi i identyfikację odchyleń. Umożliwia także symulację skutków ekonomicznych różnych wariantów decyzji, wspomaga analizę opłacalności inwestycji, ułatwia tworzenie i kontrolę wykonania budżetu oraz planów finansowych, prowadzenie rachunku kosztów i ich analizę. CSBI SA sprzedała w Polsce ponad 3 tys. licencji na produkty Progress Software Corporation.

Czcionki Bitstream Collection

Firma Bitstream przygotowała specjalny pakiet czcionek, który jest dołączany do wszystkich drukarek Hewlett-Packard sprzedawanych przez Computer 2000 Polska. Zestaw o pełnej nazwie „Bitstream Collection – wydanie specjalne dla Computer 2000 Polska i Hewlett-Packard" zawiera 250 krojów czcionek dla użytkowników Windows 3.x i 95. Czcionki zostały wybrane przez specjalistów Bitstream z biblioteki zawierającej blisko 1100 krojów. Computer 2000 dołącza pakiet do drukarek HP sprzedawanych w wielu krajach Europy.

Jaz – dysk wymienialny 1 GB

Firma Iomega, znany producent pamięci masowych, takich jak pakiet ZIP, napęd taśm Ditto Easy 3200, wprowadziła na rynek nowy napęd dysków wymiennych Jaz o pojemności 1 GB. Jaz jest przystosowany do współpracy z systemami: Windows, Windows 95, Macintosh i OS/2, a także do pracy ze stacjami roboczymi typu high-end. Urządzenie jest dostarczane wraz z pakietem oprogramowania Jaz Tools, zawierającym proste i wygodne narzędzia do katalogowania i kopiowania informacji. Napęd Jaz to niewielkie urządzenie zewnętrzne ważące 1 kg. Jest także wersja wewnętrzna. Na jednym dysku typu Jaz można zapisać: jeden film (kompresja MEG1), do 8 godzin nagrania audio o jakości CD, do 8 minut video o jakości telewizyjnej, cyfrowy album z ponad 150 zdjęciami kolorowymi o formacie 4 × 5, ponad 20 tys. zeskanowanych dokumentów.

Nowe oprogramowanie zarządzające do komputerów biurkowych oraz nowa seria płyt głównych uzupełniających elementy, tworzące platformę procesora Pentium Pro, składają się na zestaw układów 440FX Intela. Dzięki połączeniu 32-bitowej wydajności z możliwościami zarządzania oraz przystępnej cenie systemu (komputery z tym zestawem mogą kosztować poniżej 3000 USD), procesor Pentium Pro jest najodpowiedniejszy i najbardziej wydajny w przypadku zastosowań komercyjnych z wykorzystaniem Windows NT.

*

Firma 3Com wprowadziła w Europie od 1 czerwca dwa rodzaje firm dystrybucyjnych: *Volume distributors* (dystrybutorzy produktów nie wymagających dużego wsparcia technicznego) i *System distributors* (dystrybutorzy produktów zaawansowanych wymagających dużego wsparcia technicznego). W Polsce dystrybutorzy typu *Volume* to Computer 2000, MSP, TH System i Soft-tronik, zaś dystrybutorzy typu *System* to Computer 2000, Soft-tronik i System 3000.

*

Microsoft Press rozpoczął sprzedaż nowego, uzupełnionego wydania znanej książki Charlesa Petzolda „Programming Windows 95". Jest to przewodnik, który opisuje wszystkie zagadnienia związane z pisaniem programów dla najnowszej wersji Windows. Nowe wydanie książki wprowadza czytelników w świat 32-bitowego programowania. Dołączona do książki płyta CD zawiera teksty źródłowe i skompilowane wersje wszystkich programów opisywanych w książce oraz wiele przydatnych narzędzi.

*

Firma Bentley Systems tworzy MicroStation Institute, którego zadaniem jest realizacja programu profesjonalnych szkoleń w zakresie oprogramowania produkcji Bentleya. Program szkoleń zawiera m.in. kompletny plan kursów dostępnych na całym świecie w autoryzowanych ośrodkach zwanych MicroStation Productivity Center (MPC).

*

Dell Computer Corporation ogłosił, iż rozpoczyna produkcję obudów komputerowych, posiadających walory ekologiczne. Nowe obudowy będą w całości nadawały się do przeróbki wtórnej, a także, dzięki bardziej elastycznej konstrukcji, umożliwiającej znacznie dłuższy cykl rozbudowy i rekonfiguracji, będą w stanie służyć do pracy dłużej niż do tej pory.

*

Nowym produktem firmy Fujitsu jest napęd magnetooptyczny MicroMO 230 o symbolu M2541, wyposażony w złącze PCMCIA. Napęd może być dołączony do dowolnego notebooka, posiadającego takie złącze. MicroMO waży tylko 420 g, dzięki wymiennemu nośnikowi MO o pojemności 230 MB pozwala na składowanie prezentacji graficznych, multimedialnych, prac DTP czy wykonywanie standardowych kopii rezerwowych (backupu). W skład zestawu wchodzi także zasilacz sieciowy albo model M2541B przystosowany do pracy z baterii, kabel i karta do PC, jeden dysk MOD-230 MB oraz sterowniki do DOS/Win 3.x i Windows 95. Dystrybutorem produktów Fujitsu w Polsce jest firma Initel.

*

Xerox Corporation i Sun Microsystems Computer Company rozszerzyły swoją umowę o strategicznym partnerstwie. Dotyczy ona nie tylko sieciowych systemów druku na dużą skalę, ale także drukarek na biurko, co umożliwi użytkownikom produktów Suna korzystanie z kolorowych i czarno-białych drukarek Xerox. Xerox i Sun będą wspólnie wprowadzały na rynek rodzinę kolorowych drukarek laserowych Xerox XPrint oraz monochromatyczną drukarkę laserową Xerox DocuPrint 4517 mp, które są optymalnymi rozwiązaniami dla środowiska operacyjnego Sun Solaris.

*

Firma Koma S.A. z Katowic zorganizowała ogólnopolski serwis, zlokalizowany w dziesięciu miastach. Nastawiony on jest na naprawy gwarancyjne i pogwarancyjne sprzętu komputerowego firm Compaq i IBM. Koma serwisuje ponad 7 tys. stacji roboczych, 700 serwerów i kilka tysięcy urządzeń peryferyjnych (m.in. 5 tys. drukarek). Następnym etapem rozbudowy serwisu Komy będzie stworzenie serwisu aplikacyjnego baz danych (Informix).

*

Sun obniżył o 30% ceny na stacje robocze Sun Ultra. Największe zainteresowanie budzi stacja Ultra 1 model 140 z procesorem UltraSPARC taktowana zegarem 143 MHz, 32 MB RAM, dyskiem 1,05 GB, monitorem 17-calowym, grafiką TurboGX; cena tego modelu w USA spadła poniżej 10 tys. USD. Niezależnie od obniżki cen, Sun wprowadził jednoprocesorowe wersje stacji Ultra 2, które użytkownik może w razie potrzeby rozbudować, dodając drugi moduł jednostki centralnej.

*

Winterm z firmy Wyse Technology jest pierwszym w świecie terminalem dedykowanym do obsługi Microsoft Windows NT i jego aplikacji. Podłączony do serwera Windows NT (system rozszerzony o moduł WinFrame) obsługuje użytkownika Windows tak samo jak stacja PC, choć terminal jedynie wyświetla obraz i przyjmuje komunikaty z klawiatury i myszy. Winterm jest tańszy niż PC, bardziej niezawodny i bezpieczny, także w instalacjach i kosztach szkolenia. Oferowane są dwie wersje terminala: z monitorem czarno-białym i kolorowym. Obie konstrukcje są wyposażone w dwa łącza szeregowe, port równoległy, klawiaturę, mysz, oraz posiadają możliwość instalacji karty Ethernet 10BaseT lub 10Base2. Jedynym dystrybutorem tych terminali w Polsce jest firma Soft-tronik Polska.

*

Dostępne są już narzędzia Novella, umożliwiające firmom szybkie i łatwe przejście z systemów Microsoft NT Server, Microsoft LAN Manager oraz IBM LAN Server na system NetWare 4.1 i usługi katalogowe NDS (NetWare Directory Services). Narzędzia te można znaleźć w specjalnej edycji Migration Edition zestawu Novell Consulting ToolKit.

*

Oracle Corp. nabyła od Digital Equipment Corp. rodzinę produktów o wspólnej nazwie Digital's DB Integrator (DBI). W jej skład wchodzą Digital's DB Integrator, Digital's Data Distributor oraz Digital's DB Integrator Gateways. Transakcja pieniężna, której wartości nie ujawniono, będzie obejmować zarówno transfer produktów, jak również całego zespołu odpowiedzialnego za rozwój produktów i asystę techniczną dla użytkownika.

*

Firma Alantec podjęła produkcję nowej rodziny inteligentnych koncentratorów przełączających PowerHub, które posiadają jednolitą architekturę, niezależnie od ich typu i przeznaczenia. Ich wspólną cechą jest fakt, iż łączą one funkcje tradycyjnego koncentratora z funkcjami bridge'a i routera w jednym produkcie – wielowarstwowym koncentratorze przełączającym. Ich dystrybutorem w Polsce jest firma ATM.

*

W Banku Depozytowo-Kredytowym w Lublinie w maju został zarejestrowany milionowy użytkownik NetWare Novella w Polsce. Na całym świecie liczba użytkowników NetWare ocenia się na około 50 milionów. Novell oblicza, że dochody ze sprzedaży NetWare w Europie wzrosły w okresie pierwszych dziewięciu miesięcy 1995 r. o 26% w stosunku do 1994 r. Największy przyrost dochodów pochodzi ze sprzedaży NetWare 4.1, którego udział w sprzedaży przyniósł wzrost dochodów w wysokości 74% . W Polsce autoryzowani partnerzy firmy Novell sprzedali licencje dla ponad 400 tys. użytkowników systemu NetWare. Według badań firmy Dataquest, NetWare stanowił 66% rynku wśród dostawców oprogramowania dla sieci komputerowych. Na drugim miejscu uplasował się IBM, który zainstalował 13% sieci w Europie.

*

Tajwańska firma Acer (siódmy producent komputerów i akcesoriów na świecie, z obrotami wynoszącymi w 1995 r. 5,7 mld USD) zmienia w sposób znaczący swoje struktury organizacyjne. Pod bezpośrednim zarządem Tajwanu pozostanie tylko centrum badań i rozwoju oraz fabryki w Azji Południowo-Wschodniej. Zakłady produkujące na inne rynki będą ulokowane w wielu krajach na całym świecie. Działalność firmy Acer w Polsce będzie podlegała jednostce Acer Computer Austria, która ma być odpowiedzialna również za działania w Szwajcarii, Austrii oraz krajach Europy Wschodniej.

*

Warszawska firma SIMPLE udostępniła wszystkim swoim klientom oraz firmom partnerskim system pomocy technicznej obejmujący swym zasięgiem całą Polskę. Pomoc oferowana jest za pośrednictwem serwisu BBS pod warszawskim numerem telefonu (+22) 613 22 23, oraz w sieci Internet pod adresem SIMPLE@IK.ATM.COM.L.

Opracowała: Krystyna Karwicka

Nowe władze Polskiego Towarzystwa Informatycznego

PTI jest istniejącym od 15 lat stowarzyszeniem, zrzeszającym ludzi zajmujących się zawodowo informatyką i jej zastosowaniami. Obecnie PTI ma ponad 1000 członków, z których ok. 50% to pracownicy uczelni i instytutów naukowych, 30% to informatycy z przemysłu, bankowości i administracji, a pozostali to pracownicy firm komputerowych.

W Warszawie 22 czerwca odbył się V Walny Zjazd Polskiego Towarzystwa Informatycznego. Rozliczono ustępujący zarząd, wprowadzono kilka zmian do statutu stowarzyszenia i wybrano nowe władze. Jedną z ważniejszych zmian w statucie jest zmniejszenie do 50 liczby członków mogących powołać oddział terenowy PTI. Piotr Fuglewicz, ustępujący prezes PTI, za najważniejsze porażki podczas swojej kadencji uznał brak wzrostu liczebności organizacji i niezałatwienie do końca sprawy stopni specjalizacji zawodowych informatyków.

Na prezesa PTI wybrano prof. dr hab. Zdzisław Szyjewskiego. Prof. Szyjewski ukończył Wydział Matematyki na Uniwersytecie Łódzkim. Po studiach przez pięć lat pracował w Zakładzie Elektronicznych Technik Obliczeniowych w Szczecinie. Od 30 lat jest pracownikiem naukowym Uniwersytetu Szczecińskiego na Wydziale Ekonomicznym. Prof. Zdzisław Szyjewski, jest także członkiem kolegium redakcyjnego naszego miesięcznika.

W nowej strukturze władz PTI znaleźli się Piotr Fuglewicz, były prezes PTI, zastępca dyrektora CSBI, który będzie wiceprezesem ds kontaktów z zagranicą. Drugim wiceprezesem jest Jerzy Nowak, były sekretarz generalny PTI, którego zadaniami w nowym zarządzie będą kontakty z przemysłem i innymi użytkownikami technologii informatycznych w gospodarce. Jerzy Nowak jest szefem centrum informatyki w Bumar Łabędy S.A. Jarosław Deminet, dyrektor Biura Informacyjnego Kancelarii Senatu, został wiceprezesem PTI ds kontaktów z administracją i mediami. Sekretarzem Generalnym jest znany wszystkim uczestnikom internetowej listy dyskusyjnej PTI, jej organizator i operator Leszek Bogusławski, główny specjalista ds informatyki w Instytucie Ochrony Środowiska. Skarbnikiem PTI została ponownie Elżbieta Lipka.

Prezes prof. Zdzisław Szyjewski uzgodnił wstępnie z redaktorem naczelnym formy współpracy miesięcznika z PTI. „Informatyka" będzie regularnie prezentować działalność PTI na swoich łamach, a członkowie PTI będą nadal aktywnie wspierać czasopismo jako autorzy.

oprac. Lesław Wawrzonek

Książki nadesłane

Wydawnictwa Naukowo - Techniczne

Glen Waller, Paul Webster: Corel DRAW 5. Z angielskiego przełożył: dr inż. Cezary Stępień. WNT Warszawa 1996 r., wyd. I, s.336, cena 33,0 zł
ISBN 83-204-1931-X

Corel DRAW 5 jest znakomitym pakietem do rysowania i zarządzania obrazami, a jednocześnie najpopularniejszym na świecie narzędziem tego rodzaju. Używa go tysiące artystów i projektantów grafiki użytkowej. Dodatkowym atutem jest to, że pakiet ten nie wymaga jeszcze Windows 95 i można go uruchomić nawet na komputerze średniej klasy.

Książka ta stanowi przewodnik obrazkowy do programu Corel DRAW 5 - ilustracje dosłownię „pokazują", co robić, a tekst jest jasnym, zwięzłym komentarzem. Autorzy prezentują obecnie bardzo modne podejście do uczenia się programów użytkowych. W tradycyjnych książkach komputerowych znajdują się obszerne wyjaśnienia tekstowe, natomiast seria przewodników obrazkowych, użytych w tym podręczniku, uczyniła naukę programu łatwiejszą.

Użytkownik, korzystając z przewodnika obrazkowego, zaznajamia się z wyglądem ekranu i reakcjami programu. Nauka jest też szybsza dzięki temu, że nie ma rozwlekłych fragmentów tekstu. Książka jest przeznaczona dla wszystkich, którzy chcą szybko nauczyć się programu Corel DRAW 5 i wykorzystują go w praktyce.

Kay Yarborough Nelson: WINDOWS 95 - Wprowadzenie. Z angielskiego przełożyła: mgr inż. Krystyna Kietlińska. WNT Warszawa 1996 r., wyd. I, s. 147, cena 13,50 zł
ISBN 83-204-1974-3

Książka jest praktycznym wprowadzeniem dla użytkowników systemu Windows 95. Różni się on zasadniczo od systemu Windows 3.1. Obecnie używa się na przykład obu przycisków myszki. *File Manager (Menedżer plików)* i *Program Manager (Menedżer programów)* nie istnieją, a ich funkcje użytkownik wykonuje sam w różnych miejscach programu. Pozostałe różnice to między innymi : umieszczenie grup programowych w menu *Start,* umieszczanie usuwanych plików w *Recycle Bin (Kosz),* możliwość stosowania długich nazw plików, zawierających spacje, możliwość wchodzenia do różnych pracujących programów i przechodzenia od okna do okna, itd.

Wszystkie informacje, zawarte w książce, są sformułowane w taki sposób, że do przyswojenia ich nie trzeba znać wcześniejszych wersji systemu Windows. W książce znajduje się mnóstwo przydatnych wskazówek do rozwiazywania potencjalnych problemów, które może napotkać użytkownik. Zawiera ona także: spis przydatnych skrótów klawiszowych, informacje o sposobie uruchomienia i korzystania z Windows 95, opis pracy z programami użytkowymi, metody zarządzania plikami i metody przystosowania systemu do indywidualnych potrzeb.

Książka może stanowić doskonałą pomoc dla każdego początkującego użytkownika systemu Windows 95. Przejrzysta szata graficzna i dodatkowe wskazówki na marginesach sprawiają, że podręcznik jest znakomitą pomocą do korzystania z tego systemu. **(a)**

Szanowna Redakcjo

Poraził mnie pomysł Panów Fuglewicza i Miłosza dotyczący wprowadzania „informatycznych stopni zawodowych" (numer 5/1996). Jest on dowodem na to, że głęboko tkwi w wyobraźni autorów duch „komuny", która stara się reglamentować i koncesjonować wszelką możliwą działalność. Przykład stopni specjalizacyjnych dla lekarzy nie jest ani adekwatny, ani zachęcający.

Po pierwsze: Facet, który ma wyciąć wyrostka lub wstrzyknąć truciznę, musi niestety posiadać jakieś „papiery", że zrobi to poprawnie. Są one potrzebne choćby tylko jako ochrona prawna tych niebezpiecznych czynności. Przykład Korvity wskazuje na to, że i bez tych „papierów" można leczyć całkiem nieźle (ale i mieć kłopoty prawne).

Po drugie: Ceną, jaką płaci się w medycynie za ten sposób nobilitacji zawodowej, jest atmosfera protekcji, a często i korupcji, którą (niestety) obrósł ten system. Izby lekarskie stanowią w istocie lobbies, które często celowo ogranicza dopływ młodzieży do intratnych specjalności.

Po trzecie: Sytuacja specjalizacji inżynierskich wskazuje na to, że nawet w latach PRL-u życie broniło się przed koncesjonowaniem i reglamentacją tam, gdzie to nie jest konieczne. Praktyczny sens mają tylko uprawnienia zawodowe w specjalizacjach, gdzie odpowiedzialność jest podobna jak w medycynie.

Po czwarte: Przywoływanie przykładów zachodnich, a zwłaszcza angielskich, do niczego nie prowadzi. Wszędzie są socjaliści, którzy dążą do wzrostu roli państwa w każdej dziedzinie. Wiadomo, że pomysły laubourzystów w Wlk. Brytanii niewiele odbiegały od PRL-owskich, tylko że na szczęście nie wprowadzili ich w życie, a pani Thatcher doprowadziła system „do normy".

Po piąte: „Komputerowe prawa jazdy" to już zupełnie ponury żart. Co innego jest prawo do prowadzenia samochodu, którym można spowodować wypadek, a nawet śmierć człowieka. Natomiast wymaganie „prawa jazdy" od gościa, który w Wordzie chce napisać list do narzeczonej – to po prostu śmieszne! Jeśli panowie autorzy powiedzą, że przy pomocy sieci i komputera można narobić szkód w bazach danych – to od razu odpowiadam: tacy hackerzy napewno zdobędą wymagane „prawo jazdy".

At last but not least: Rozumiem, że Pan Fuglewicz z Towarzystwa Informatycznego może kierować się korzyścią tego Towarzystwa (które zapewne chce uczestniczyć w tym koncesjonowaniu) – „niezła fucha". Jaką korzyść mają z tego pomysłu odnieść informatycy – doprawdy nie wiem. Po stokroć nie!!!

Z głębokim poważaniem

Doc. dr inż Krzysztof Gosiewski
Instytut Chemii Nieorganicznej
ul. Sowińskiego 11
44-101 Gliwice

Szanowna Redakcjo

List docenta Gosiewskiego napełnił mnie głębokim zachwytem. Pomijając niezrozumiałe fragmenty dotyczące jakichś spraw politycznych, którymi o ile pamiętam nie zajmowaliśmy się z kolegą Miłoszem w tekście, na który list był reakcją, jest on wspaniałą egzemplifikacją pewnej siebie ignorancji, której stopnie zawodowe mają postawić tamę. Abstrahując od słowa na „k", którego Polacy z wykształceniem od niepełnego wyższego wzwyż używają w miejsce innego – również na „k" choć krótszego – używanego przez prosty lud, postaram się odnieść do meritum wypowiedzi Pana Docenta.

A więc – po pierwsze: „facet, który ma wyciąć wyrostka" (tak w oryginale) nie musi mieć żadnych papierów bo to się nazywa „mokra robota" i raczej nie podlega rejestracji. Natomiast lekarz wycinający wyrostek **oprócz** studiów teoretycznych musi wykazać się doświadczeniem praktycznym. Jest tak dlatego, że konsekwencje błędów lekarza mogą być nieodwracalne. Chodzi o ludzkie życie. Podobnie sprawa ma się w przypadku błędów związanych z wprowadzeniem informatyki. Koszt politycznej decyzji (Berliet bis) podjętej wbrew opinii kilku profesjonalnych ekspertów w sprawie wyboru sposobu realizacji informatycznego systemu obsługi podatków – dziś znanego jako POLTAX – wynosi biliony starych złotych nieściągniętych podatków, a pośrednio (choć można takie stwierdzenie nazwać demagogią) życie osób, dla których zabrakło w szpitalach leków, które można byłoby za te pieniądze kupić, gdyby były.

Po drugie: nie mam zdania, nie znam się na medycynie co najmniej w tym samym stopniu co doc. Gosiewski na informatyce, ale mam tyle śmiałości, żeby się wypowiedzieć. Od żony dowiedziałem się, że Izby Lekarskie nie mają nic wspólnego ze specjalizacjami lekarskimi. Po trzecie: Zgadzam się – odpowiedzialność w informatyce, bywa porównywalna z tą, jaka występuje w medycynie. Od poprawności działania niektórych programów może zależeć życie ludzkie. Przykłady są tak liczne i tak oczywiste, że aż nie warto ich przytaczać.

Po czwarte: nie wiem co ma do rzeczy brytyjska Partia Pracy. Z tego co wiem nie mieszała się do sprawy. Patronem działalności BCS jest Książe Kentu. Podziwiam zarówno Margaret Thatcher, jak i erudycję Pana Docenta. Ja osobiście wykorzystałbym jeszcze fakt istnienia uprawnień budowlanych w powiązaniu z działalnością lóż wolnomularskich. Posądzenie o udział w sprawie masonerii, z niewielką przymieszką homoseksualizmu na pewno dodałoby tekstowi pieprzu.

Po piąte: nigdzie w naszym artykule nie jest napisane, że komputerowe prawo jazdy ma być obowiązkowe. Wina oczywiście moja, bo pewnie niejasno napisałem, ale Pan Docent nie zrozumiał. Ma ono w skali całej Europy służyć potwierdzeniu elementarnych umiejętności obsługi komputera, i jako takie może być przydatne w trakcie starań o pracę lub o pracownika. Hakerzy tu „ni pricziom" (zechce Pan Docent zwrócić uwagę na ewidentny rusycyzm).

Wreszcie po last but not least: Trafnie odczytał Pan ten aspekt zagadnienia. Wprowadzenie stopni specjalizacyjnych ma służyć interesowi Polskiego Towarzystwa Informatycznego na co najmniej dwa sposoby. Po pierwsze: grono ludzi kompetentnych, o potwierdzonych osiągnięciach naukowych i zawodowych mogłoby w ten sposób uzyskać faktyczny (choć pośredni) wpływ na ważne decyzje związane z informatyką. Po drugie: dzięki konsekwentnemu wprowadzeniu stopni zawodowych mógłby się podnieść ogólny poziom informatyki w Polsce, którego podnoszenie jest statutowym celem Towarzystwa.

Na koniec po ostatnie: informatyka jest zarówno nauką, jak i sztuką inżynierską. Obok medycyny jest dziedziną, w której stałe i bardzo pracochłonne uzupełnianie kompetencji jest bezwzględną koniecznością. Skoro nie można zbudować legalnie altanki i założyć w niej elektryczności bez co najmniej dwóch panów: jednego z uprawnieniami budowlanymi, a drugiego z elektrycz-

nymi, to tym bardziej naturalne wydaje się, że dziedzina o złożoności (z całym szacunkiem dla wyżej wymienionych) wielokrotnie większej powinna również mieć swój system weryfikacji umiejętności zawodowych. Resztę napisaliśmy w artykule.

Nie znam osobiście pana Gosiewskiego. Internet powiedział mi tylko, że na dwunasty międzynarodowy kongres poświęcony procesom inżynierii chemicznej CHISA'96, który odbędzie się w Pradze w dniach 25–30 sierpnia 1996 przyjęto mu poster: „Operator supporting computer expert system based on mathematical models and industrial data processing for metallurgical sulphuric acid plant", którego współautorami są mój dawny kolega ze studiów na jednej z najlepszych uczelni kształcących informatyków – Politechnice Śląskiej – Z. Ogonowski oraz R. Marczycki z bardzo porządnej firmy DST, zarządzanej przez K. Sromka, z którym kiedyś pracowałem i którego umiejętności praktyczne cenię wysoko. Tak więc z punktu widzenia informatyki obraca się Pan Gosiewski w bardzo dobrym towarzystwie. Nie sądzę, żeby był zachwycony, gdybym któregoś dnia zjawił się u Niego w instytucie i zażądał udziału w projektowaniu technologii procesów chemicznych na podstawie sukcesu, jakim w młodości było wypędzenie wszystkich z pracowni chemicznej, w wyniku skutecznego wytworzenia siarkowodoru.

Z wyrazami szacunku dla chemików
i prośbą o podobne potraktowanie mojej profesji
Piotr Fuglewicz
informatyk

PS. A tak naprawdę najważniejszym problemem obecnych dni są próby zawłaszczenia samorządu gospodarczego przez Państwo, zawarte w projekcie odnośnej Ustawy. „Praworządność socjalistyczna", „centralizm demokratyczny" i „obowiązkowy samorząd" to pojęcia z tej samej poetyki. Atakowany przez Pana Docenta projekt ma w odróżnieniu od wyżej wspomnianego charakter uczciwie samorządowy. Piszę o tym, bo trzeba dziś wszędzie krzyczeć na ten temat i żeby potem nie było, że ja byłem za.

Szybkie wytwarzanie aplikacji na Wiosennej Szkole PTI

Leszek Bogusławski
Zdzisław Szyjewski

Kolejna edycja Wiosennej Szkoły PTI poświęcona była metodom szybkiego wytwarzania aplikacji – Rapid Application Development (RAD). Szkoła tradycyjnie odbyła się w Świnoujściu w ośrodku Dolna Odra, 13–17 maja. Wydaje się, że tematyka została dobrze dobrana, o czym może świadczyć udział ponad stu uczestników. Aktywność w trakcie obrad i frekwencja na sali potwierdzają tę tezę. W programie Szkoły było 10 prelekcji, a właściwie wykładów poświęconych organizacji, narzędziom i metodom stosowanym w metodyce RAD. Przybyli również główni dostawcy krajowi, którzy zajmują się tą tematyką.

Zgodnie z kilkuletnią tradycją firmę Rodan System prezentował jej prezes Witold Staniszkis. Przemawiał treściwie, więc jeśli by ktoś chciał wyjaśnić dokładnie wszelkie pojęcia, którymi dr Staniszkis operował, to powstałby wykład-rzeka z zagadnień baz danych, ze szczególnym uwzględnieniem nowych technologii.

W sesji popołudniowej tego dnia pracownik Rodan System Marek Łubiński prezentował wykład na temat nowego produktu Jasmine, który ma być w najbliższym czasie wprowadzony na rynek. Dzięki tej prezentacji uczestnicy dowiedzieli się o kierunku prac jednego z największych dostawców oprogramowania.

Drugi dzień również tradycyjnie od kilku lat rezerwuje CSBI z Progressem. Tym razem Jarosław Kaczorowski i Jerzy Sitarz przedstawili wersję 8.0 systemu i jej walory w aspekcie szybkiego wytwarzania aplikacji.

W sesji popołudniowej Wiosennej Szkoły debiutował System 3000. Firma przedstawiła pakiet SAPIENS, który jest klasycznym przykładem zastosowania metodyki RAD.

Dzień trzeci to trzy spotkania, czyli zajęcia programowe do 21.00. Rano Tomasz Kuźmierz i Grzegorz Bartler z SAS Institute, przedstawili prelekcję na temat RWT (Rapid Warehousing Technologies), wzbogaconą prezentacją działania produktu.

Po południu Polnet Technologies prezentował UNIFACE SIX. Grzegorz Prochowski doskonale pod względem dydaktycznym i merytorycznym wprowadzał uczestników w tajniki systemu do szybkiej budowy aplikacji klient/serwer drugiej generacji. Prezentacja przybliżyła obecnym kompletne podejście metodyczne z narzędziami do metodyki RAD.

Wieczorem zostaliśmy przyjęci przez InfoVIDE. Firma ta przedstawiła swój punkt widzenia na problem szybkiego wytwarzania aplikacji. Panowała miła atmosfera, przeprowadzono nawet konkurs z nagrodą w postaci PowerBuildera.

Czwartego dnia występował Krzysztof Mikołajewski z INFORMIX Polska, który dokonał przeglądu najnowszych produktów firmy i udzielał wskazówek, jak migrować do NewEra 3.0.

Po południu Mariusz Gibułtowski i Grzegorz Wojarnik, młodzi pracownicy Uniwersytetu Szczecińskiego, mówili na temat DELPHI 2.0. Prezentacja ta nie tylko w formie, ale i w treści różniła się od innych. Wynikało to z faktu, że produkt prezentowali pracownicy naukowi, którzy nie musieli dbać o efekty marketingowe i mogli skoncentrować się na rzeczywistych wadach i zaletach produktu.

Od kilku lat tradycją Szkoły Wiosennej dzień piąty i ostatni to prezentacja IBM. Tym razem nie była to prezentacja zgodna z tematyką Szkoły. Krzysztof Chełpiński z IBM zaprezentował bowiem wykład na temat: dlaczego jego firma szuka partnerów, w jakim celu i na jakich warunkach. Spotkanie z IBM trwało ponad 3 godziny, lwia część z tego miała charakter rozmów bilateralnych kandydatów na przyszłych partnerów z szefem handlowym tej firmy.

Z punktu widzenia organizatorów można powiedzieć, że impreza się udała. Program został zrealizowany w całości, zgodnie z przygotowanym terminarzem. Frekwencja uczestników na zajęciach była bardzo wysoka, co świadczy o zadowalającym poziomie prelekcji. Dyskusje w trakcie obrad oraz spotkania pozasesyjne spełniły oczekiwania organizatorów, czyli nie tylko dostarczyły dużą dawkę wiedzy, ale również zintegrowały uczestników. Uczestnikami byli w dużej części członkowie PTI, bo głównie dla nich organizowane są Wiosenne Szkoły. Mimo, że akcja propagandowa prowadzona jest wśród członków PTI, licznie przybywają również sympatycy Towarzystwa.

Nagrody targowe

Rada Programowa IV Międzynarodowych Targów Zastosowań Informatyki w Przemyśle i CAD/CAM'96 przyznała firmom, które wzięły udział w imprezie, nagrody i wyróżnienia w trzech kategoriach.

- **Za opracowanie i wdrożenie polskiej technologii** - nagrodę otrzymała firma informatyczna **RoboBAT** - autor programu Robot V6 będącego oryginalnym systemem metody elementów skończonych. Wyróżnienie otrzymała firma **Inter-Design** za pakiet programów do obróbki rastrowej i wektorowej dokumentacji.
- **Za promocję technologii informatycznej** - nagroda przypadła firmie **Visitronics**, propagatorowi informatyki w przemyśle, budownictwie i administracji. Wyróżnienie otrzymał **Zarząd Miasta Szczecin**, za promocję rozwiązań informatycznych w zarządzaniu miastem, planowaniu przestrzennym i prognozowaniu infrastruktur miejskich oraz projekt „Bezpieczne miasto".
- **Za najbardziej kompleksową ofertę dla przemysłu** - nagrodę przyznano firmie **Hewlett-Packard**, za kompleksową, zintegrowaną ofertę obsługi przemysłu. Wyróżnienie w tej kategorii otrzymał **Centralny Ośrodek Informatyki Górnictwa**, za kompleksową obsługę gmin i górnictwa.

Stowarzyszenie Użytkowników Systemów Inżynierskich przyznało swoją nagrodę **Służbie Informatycznej Zakładu Lotniczego PZL Mielec** - za całokształt działalności w opracowywaniu systemów CAD/CAM i wdrażaniu technologii komputerowej w wytwarzaniu samolotów, samochodów, satelitów komercyjnych i oprzyrządowania. SUSI przyznało też dwa wyróżnienia. Otrzymały je: przedsiębiorstwo **Prozap** z Puław za wprowadzenie nowoczesnych światowych technologii CAD, modelowanie procesów technologicznych w przemyśle chemicznym oraz sprzedaż kompleksowych projektów inżynierskich na całym świecie; **Instytut Obróbki Skrawaniem** z Krakowa - za pakiet oprogramowania służącego do automatycznego programowania obrabiarek sterowanych numerycznie, uwzględniającego możliwości polskiego przemysłu.

Nagroda publiczności w postaci samochodu Polonez Caro ufundowanego przez Daewoo-FSO przypadła w udziale firmie **Intergraph**, za produkt **Solid Edge**, umożliwiający projektowanie części i zespołów mechanicznych na komputerach osobistych, przy użyciu środowiska Windows 95 lub Windows NT. Nagroda została przekazana na cele społeczne. (k)

Informatyka dla przemysłu

Lesław Wawrzonek

W Warszawie zakończyły się „Międzynarodowe Targi Zastosowań Informatyki w Przemyśle i CAD/CAM - 96". Patronat nad nimi sprawowało Ministerstwo Przemysłu i Handlu oraz Stowarzyszenie Użytkowników Systemów Inżynierskich - SUSI.

Chociaż Bogusław Dziarnowski, pełnomocnik ds informatyki ministra przemysłu i handlu, główny animator Targów, każdego rozmówcę zapewniał, że są to także targi zastosowań informatyki w przemyśle, a nie tylko CAD/CAM, to jednak każdemu zwiedzającemu rzucała się w oczy ilościowa przewaga oprogramowania i sprzętu dla CAD/CAM/CAE nad oprogramowaniem do zarządzania produkcją klasy MRP i systemami finansowo-księgowymi, kadrowymi czy sprzedażą. Wydaje się, że formuła targów zawężona do sprzętu i oprogramowania CAD/CAM/CAE byłaby w zupełności wystarczająca, aby zapełnić pałacowe sale wystawcami i liczną rzeszą zwiedzających je specjalistów.

Zdaniem stałych bywalców tych Targów widać wyraźny wzrost zainteresowania nimi zarówno firm, jak i zwiedzających. Wynika to nie tylko z utrwalania się tradycji targów, ale chyba przede wszystkim z powodu wzrostu gospodarczego, który zawsze sprzyja inwestowaniu. W warunkach rosnącej konkurencji nowoczesne systemy informatyczne stają się podstawowym warunkiem umożliwiającym nie tylko nadążanie za potrzebami rynku, ale wręcz utrzymanie się na nim. Rzeszowski Zelmer stanowi jeden z najbardziej spektakularnych przykładów sukcesu produkcyjnego, wynikającego w dużym stopniu z zastosowania informatyki w projektowaniu.

Na Targach nie były praktycznie reprezentowane instytucje naukowe. Brak zainteresowania przemysłu ofertą instytucji naukowych nie jest, jak widać, winą pracowników przemysłu. Po prostu - albo polska nauka nie ma nic do zaoferowania albo nie potrafi swoich możliwości zaprezentować. Skoro jednak siedem instytucji naukowych i uczelni miało coś do pokazania na wystawie CeBIT'96, to dlaczego nie były one reprezentowane na targach w Warszawie?

Dużym atutem Targów, choć nie w pełni wykorzystanym przez organizatorów, były liczne specjalistyczne seminaria, zgrupowane w kilku blokach programowych. Przeglądając plan seminariów, bo nie sposób wysłuchać prawie stu w ciągu czterech dni, można znaleźć zaledwie kilka przeglądowych wykładów pracowników nauki. Prezentacje produktów, choć cenne, dają niewielkie pojęcie uczestnikom o wadach, zaletach i ograniczeniach przedstawianego sprzętu, oprogramowania czy kompleksowych rozwiązań. Organizatorzy powinni zadać sobie pytanie: kto może zapłacić za przeprowadzenie uczciwej i głębokiej analizy określonej klasy produktów i przedstawienie jej potencjalnym klientom - gościom targów? Powinni się też zastanowić nad tym, że na kieleckiej konferencji „Metoda MRP II w przemyśle" tylko dwa wystąpienia nagrodzono brawami. Były to wystąpienia pracowników przemysłu, dzielących się swoimi doświadczeniami.

Należy życzyć organizatorom Targów, aby kolejne były coraz lepsze oraz aby nie zaliczyli do swoich błędów decyzji o bezpłatnym wstępie dla wszystkich zainteresowanych.

I Ogólnopolskie Zawody w Programowaniu Zespołowym

Polskie Towarzystwo Informatyczne – Koło w Poznaniu – Instytut Informatyki Politechniki Poznańskiej i Koło Naukowe Studentów Informatyki BOOBOO, przy wsparciu MEN organizują w Poznaniu w dniach 25–26 października br. I Ogólnopolskie Zawody w Programowaniu Zespołowym. Jest to okazja do zademonstrowania i doskonalenia umiejętności w zespołowym rozwiązywaniu problemów informatycznych. Zawody umożliwią również nawiązanie kontaktów między studentami z różnych ośrodków akademickich, pozwolą zapoznać się ze specyfiką tego rodzaju rywalizacji oraz pomogą przygotować się do ewentualnego uczestnictwa w ACM International Collegiate Programming Contest. Każda drużyna będzie miała do dyspozycji jeden komputer i pięć godzin na rozwiązanie jak największej ilości zadań z zestawu 6–8 zadań przygotowanych przez organizatorów. Rozwiązania muszą być przedstawione w języku C zgodnym z normą ANSI. Tak jak w zawodach organizowanych przez ACM, będą one sformułowane w języku angielskim. Zawodnicy będą mogli korzystać z przyniesionych przez siebie książek, notatek, słowników, itp. Niedozwolone natomiast będzie korzystanie z kalkulatorów, notebooków i innego sprzętu elektronicznego.

Zawody mają trwać dwa dni i odbędą się w Instytucie Informatyki Politechniki Poznańskiej. Pierwszy dzień zawodów (25.10.96) przeznaczony jest na aklimatyzację uczestników. Wtedy odbędzie się sesja wykładowa, a po niej próbna, w czasie której uczestnicy będą mogli zapoznać się ze środowiskiem programistycznym, w jakim przyjdzie im pracować (sieć komputerowa, tryb oceniania zadań, itp.). Właściwa część zawodów rozpocznie się w sobotę 26 października o godz. 10^{00}, a zakończy o godz. 15^{00}. Po zakończeniu zawodów około godziny 15^{30} zostaną przedstawione oficjalne wyniki.

Uczestnicy zawodów

Każda drużyna biorąca udział w zawodach składa się maksymalnie z trzech osób – studentów informatyki na polskich uczelniach. W zawodach nie mogą brać udziału osoby, które już ukończyły studia informatyczne w kraju lub za granicą, ani te, które dwukrotnie startowały w finałach ACM International Programming Contest. Przynajmniej dwóch członków drużyny musi być studentami co najwyżej czwartego roku studiów. Przedstawiciel wydziału, na którym studiują uczestnicy, musi zaświadczyć, że spełniają oni warunki konkursu. Każda drużyna powinna wyznaczyć osobę, której zadaniem będzie kontaktowanie się z organizatorami (może to być członek drużyny). Każda uczelnia prowadząca studia na kierunku informatycznym ma prawo wystawić jeden zespół. Uczelnia może zgłosić większą liczbę zespołów, które zostaną zakwalifikowane do wzięcia udziału w zawodach, w przypadku wolnych miejsc.

Zasady oceniania

Drużyny będą oceniane na podstawie liczby poprawnie rozwiązanych zadań. W przypadku rozwiązania tej samej liczby zadań, o kolejności zadecyduje łączny czas rozwiązywania. Jest on sumą czasów potrzebnych na rozwiązanie poszczególnych zadań (od początku zawodów do momentu przedstawienia poprawnego rozwiązania). Zadania będą oceniane na bieżąco i jeśli rozwiązanie nie zostanie zaakceptowane, drużyna będzie mogła przedstawić poprawną wersję. Za każde niepoprawnie przedstawione rozwiązanie drużyna dostanie 20 minut kary. Kara jest naliczana dla każdego zadania osobno, a łączny czas rozwiązywania zadań będzie obliczany tylko na podstawie zaakceptowanych rozwiązań.

Zgłaszanie zadań

Organizatorzy zwracają się do pracowników wyższych uczelni prowadzących studia na kierunku informatycznym z prośbą o nadsyłanie propozycji zadań. Należy je przesyłać na dyskietce 3.5", zawierającej pliki w formacie systemu MS DOS (tekst ma być reprezentowany w postaci ciągów znaków ASCII). Zgłoszenie powinno zawierać:

- tekst zadania w języku angielskim: sformułowanie problemu, opis postaci danych wejściowych, opis postaci wyników, przykład bądź przykładu działania programu,
- plik „task.txt",
- rozwiązanie (opis rozwiązania oraz program w języku C),
- pliki „comments.txt" oraz „task.c",
- szereg testujący: dane wejściowe dla pierwszego testu i spodziewane wyniki dla tego testu, dane wejściowe dla drugiego testu i wyniki, itd.,
- pliki „1.in", „1.out", „2.in", „2.out", ...
- dane o autorze: imię i nazwisko, instytucja, adres pocztowy i elektroniczny, telefon, faks,
- plik „personal.txt".

Propozycje zadań należy przesyłać najpóźniej do 2 września 1996 r. na adres: **Jerzy Nawrocki, Instytut Informatyki, Politechnika Poznańska, ul. Piotrowo 3a, 60–965 Poznań, e-mail: nawrocki@pozn1v.put.poznan.pl.**

Autorzy zaakceptowanych zadań będą proszeni o sędziowanie zawodom i otrzymają honorarium. Przykład zgłoszeń można znaleźć na stronie http: //www.cs.put.poznan.pl/booboo/zawody/

Sprawy organizacyjne

Imienne listy zawodników można zgłaszać do 15.10.1996 r. Ostateczne potwierdzenie uczestnictwa będzie przesłane do 18.10.1996 r.

Komitetowi programowemu przewodniczy prof. Jan Węglarz z Politechniki Poznańskiej, przewodniczącym Komitetu Organizacyjnego jest mgr Cezary Laskowski, który prowadzi też rejestrację zawodników.

Rada w radę...

dokończenie z I s. okł.

Na początek zalecenia

Departament Informatyki RM i Rada Koordynacyjna ds. Informatyki przy RM są zdecydowane lepiej zadbać o poziom kadr informatycznych w administracji. Na razie postawiono diagnozę i sformułowano wstępne zalecenia. Nie ma jeszcze harmonogramu, nakładów i terminów, czyli tzw. studium wykonalności, które mają powstać po zatwierdzeniu strategii przez Rząd.

Jednym z zamierzeń jest podniesienie płac informatyków, co musi spowodować przełamanie dotychczasowych schematów wynagradzania, obowiązujących w administracji. Równocześnie planuje się zwiększenie wymagań formalnych co do poziomu kwalifikacji poprzez stworzenie struktury stopni specjalizacji zawodowej. Od wielu lat zabiega o to Polskie Towarzystwo Informatyczne. Omówienie tego projektu przez Piotra Fuglewicza i Marka Miłosza przedstawiliśmy w numerze 5/96 Informatyki.

Dr Wacław Iszkowski, prezes Polskiej Izby Informatyki i Telekomunikacji jest sceptyczny co do szybkiego powodzenia tego projektu. Twierdzi on, że „jest zbyt wcześnie, aby udało się wprowadzić stopnie specjalizacji dla informatyków, gdyż środowisko to nie jest jeszcze dostatecznie skonsolidowane". Jako przykład do naśladowania podał ekonomistów, którzy jako grupa zawodowa dojrzeli do tego, aby bronić swoich dochodów i statusu ograniczając wszystkim chętnym dostęp do zawodowo prowadzonej rachunkowości. Tak więc jest całkiem możliwe, że już niedługo świadczenie usług prowadzenia ksiąg rachunkowych kilku firm będzie wymagało zdobycia certyfikatu, natomiast do prowadzenia projektów wartych miliony dolarów nadal będą wystarczały dobre chęci.

Departament Informatyki RM dostrzega też walory pośrednich metod pozyskania wysoko kwalifikowanej kadry informatycznej. Będzie dążył do zatrudniania wykładowców zagranicznych na kilkuletnich kontraktach w dziedzinach, w których nie ma u nas zbyt wielu specjalistów. Mają być też dotowane specjalistyczne, a tym samym niskonakładowe książki informatyczne.

Nie zdają egzaminu – organizowane dotychczas bez planu – szkolenia pracowników państwowych zajmujących się informatyką lub tylko korzystających z jej narzędzi. Rządowe informatyczne centrum szkoleniowe, podobnie jak to jest w innych specjalnościach, ma rozwiązać problem szkoleń w administracji.

Z omówionego tu fragmentu raportu „Strategia Informatyzacji Administracji Państwowej" dotyczącego kadr widać, że problemy jakie dostrzega Departament Informatyki URM i metody ich rozwiązania są zgodne z tym, co sądzą postronni obserwatorzy działalności Rządu w tym zakresie. Najważniejsze jest jednak to, czy i w jakim stopniu uda się te zamierzenia zrealizować. Będziemy obserwować, informować i komentować.

Sobota? - odpada

Leszek Bogusławski

W naszym mieście odbyło się spotkanie WAŻNEGO GREMIUM (WG). Jego celem było omówienie i przygotowanie się do zebrania INNEGO WAŻNEGO GREMIUM (IWG). INNE WAŻNE GREMIUM miało wybrać delegatów na spotkanie BARDZO WAŻNEGO GREMIUM NASZEGO GRONA (BWGNG). No i cóż, zebranie jak zebranie, ot nic specjalnego. Atmosfera była rzeczowa. Trudno by jednak doszukiwać się w niej ciepła, przyjaźni, raczej chłodu, choć koleżeństwa. W końcu nie o atmosferę tu idzie, a o cel, a ten osiągnięto. Ustalano termin niezwykle istotnego dla IWG zebrania, które miało wyłonić nowe WG. Takie zebranie odbywa się bardzo rzadko: raz na trzy, no może cztery lata. Śmiało można stwierdzić, iż jest to najważniejsze zebranie IWG. Propozycje terminu padały szybko. Kierowano się przy tym daleko posuniętym realizmem. Pierwszą propozycję zbito faktem mogących się odbyć *którychś-tam* targów *czegoś-tam*. Padła druga: sobota. *No nie* - mówi jeden przedstawiciel WG - *mnie to się nie chce ruszać w sobotę, odpada. No właśnie* - dorzucił drugi - *sobota jest kiepska, a niedziela to działka. Otóż to* - rzekł trzeci - *piątek to też przed weekendem, więc na pewno wszyscy wyjadą.*

Na szczęście są dni „międzyweekendowe", więc ustalono środek tygodnia po południu. Wprawdzie w końcu tygodnia byłoby więcej czasu na rzetelną dyskusję, ale tak to przynajmniej będzie krótko (jak to po pracy) i treściwie, a o to przecież chodzi.

Pierwszy raz słuchałem przedstawicieli WG. Przez chwilę chodziło mi po głowie pytanie: czy oni nie są tu przypadkiem za karę?

Zebranie zostało zamknięte. Poszliśmy na spotkanie IWG. Było nas teraz dwukrotnie więcej. Po wszystkich wstępach, przywitaniach, owacjach przystąpiono do meritum. Rozpoczęto od dyskusji, pozostawiając na koniec sprawę wyborów kandydatów do BWGNG. Krótko mówiąc - skoncentrowano się na próbie krystalizacji stanowiska, które ma być prezentowane na BWGNG. Wszyscy, jak jeden mąż, wypowiadali swoje zdanie. Wypowiedzi były różne, głównie odnosiły się do zaszłości, ale dużo było propozycji nowych kierunków działalności. Także od niedawnych uczestników WG dowiedziałem się, co i jak trzeba robić, aby w NASZYM GRONIE było wspaniale. To dobrze, bo człowiek uczy się całe życie. Pomysły sypały się jak z rękawa, tylko siąść i zapisywać. Oczywiście okraszone to było krytyką działalności poprzedniego WG, tym razem tego wybieranego na spotkaniu BWGNG. Brakowało w nich tylko jednego, choć „z pozoru" bardzo istotnego elementu. Odpowiedzi na pytanie: kto mianowicie ma to wszystko robić? Nie, nie, to chyba za ostro powiedziane, raczej jakie stworzyć warunki, by można było doprowadzić do realizacji choćby 10-tej części tych postulatów. Przepraszam, chyba jednak nie stać mnie na dyplomację. Najbardziej pasowałoby tu coś, co w wolnym tłumaczeniu brzmi: jak to zrobić by zarobić, a nic, w ogóle nic, nie robić. Aż strach pomyśleć, jeśli inni delegaci IWG przyjadą na spotkanie BWGNG z takimi propozycjami, to ...

Jestem głęboko przekonany, że jednak przyszłość przed NASZYM GRONEM maluje się w jasnych barwach. W końcu jesteśmy w nim, bo taka była nasza wola, i choć na pierwszy rzut oka wygląda to nieciekawie, to myślę, że w tak dużym GRONIE znajdzie się kilku takich, co to im się będzie chciało... (przyjść w sobotę).

Kontakt z autorem:
e-mail: jerome@warman.com.pl

Zaprosili nas...

Novell na konferencję prasową oraz uroczystość związaną z przyznaniem dyplomu dla milionowego użytkownika NetWare oraz na cykl prezentacji „Connecting YOUR World", dotyczący najnowszych trendów i technik tworzenia, przesyłania i dostępu informacji.

iomega na konferencję prasową, podczas której zaprezentowane zostało nowe oblicze firmy oraz kierunki jej rozwoju.

Microsoft na konferencję prasową towarzyszącą premierze systemu wymiany i organizacji informacji *Microsoft Exchange*.

MMI AWATAR S.C. (polskie przedstawicielstwo **Messe München GmbH**) na prezentację prasowo-branżową, poświęconą prezentacji imprez targowych, odbywajacych się na Terenach Targowych w Monachium: **SYSTEMS '96** i **electronica '96**.

BUiK „GAMBIT" i Centralny Ośrodek Informatyki Politechniki Warszawskiej na seminarium, którego celem była prezentacja programu *Mathematica®*.

IV Międzynarodowe Targi Zastosowań Informatyki w przemyśle i CAD/CAM '96 na ceremonię oficjalnego otwarcia targów oraz na uroczystość wręczenia nagród targowych.

Szczeciński System Informacji Geograficznej na swoje stoisko na targach CAD/CAM '96 oraz na seminarium w ramach bloku tematycznego „Miasto i gmina w komputerze" - „Szczecińska koncepcja SIG".

FS Poland Sp. z o.o. na swoje stoisko na targach CAD/CAM' 96 oraz na seminarium.

Intergraph Europe (Polska) na inaugurację polskiej wersji programu Solid Edge

Computerland na seminarium dotyczące sieci rozległych (Corporate Networks).

ATM do swojego stoiska na targach CAD/CAM '96 (prezentacja komputerów Silicon Graphics, oprogramowania firmy Alias/Wavefront i najnowszego systemu sieciowego firmy FORE Systems) oraz na seminaria.

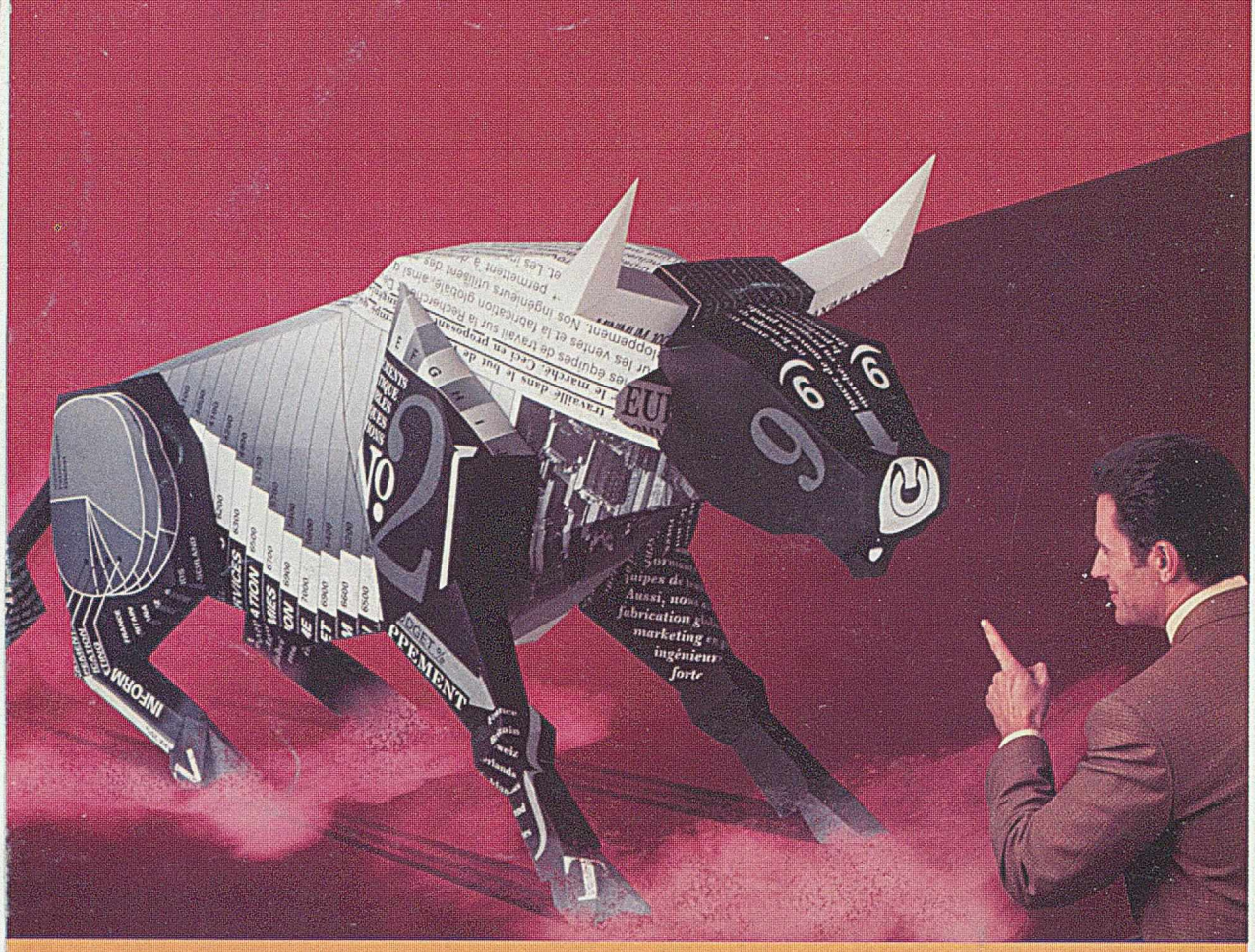
ATM do swojego stoiska na ComNet '96 (prezentacja najnpwszego sprzętu sieciowego, m.in. firmy Cabletron Systems) oraz na seminaria towarzyszące wystawie.

Dell Computer Poland Sp. z o.o. na uroczystą kolację z okazji czwartej rocznicy działalności firmy.

DEUTSCHE MESSE AG, HANNOVER oraz Ewa Samkowska, przedstawiciel w Polsce, na konferencję prasową CeBIT HOME'96.

PROGRESS na seminarium poświęcone prezentacji środowiska projektowego - Progress v 8. **(a)**

JUNISOFTEX
20 LAT DOŚWIADCZEŃ WDROŻENIOWYCH
Perfect-Ekspert
Sieciowy i zintegrowany system informatyczny pracujący na platformach PC i AS/400, głównie dla dużych firm produkcyjnych i handlowych.
Przystosowany do elektronicznej wymiany dokumentów (EDI-zamówienia, faktury, WZ itp.).
wielowalutowości oraz rachunkowości zarządczej.
Wyposażony w system ochrony danych generowanych wewnątrz i na zewnątrz przedsiębiorstwa.
Analizy Marketingowe
System Finansowo Kosztowy
System Obrotu Towarowego i Fakturowania
System Informacji Kierownictwa
System Gospodarki Materiałowej
System Przygotowania, Planowania i Rozliczania Produkcji
System Środków Trwałych i Wyposażenia
System Kadry-Płace
Monitorowanie Procesów Produkcji, Kontrola Jakości
Inżynieria Wiedzy Systemy Ekspertowe
Inne Systemy Ewidencyjne
Inteligentny System Analiz Finansowych
Oferujemy kompleksowe usługi w zakresie:
- doradztwa organizacyjnego, merytorycznego i technicznego
- organizacji, projektowania i wykonawstwa innych systemów informatycznych
- wdrażania, konserwacji i modernizacji systemów informatycznych
- projektowania, wykonawstwa dowolnego typu sieci komputerowych, z uwzględnieniem już istniejących rozwiązań
- dostawę sprzętu komputerowego i sieciowego
- szkoleń użytkowników w zakresie operatora i administratora systemu AS/400, jak również obsługi pakietów programowych PC Support 400 lub Client Acces /400 przeznaczonych do współpracy z PC
Elektrownia RYBNIK
Zakłady Rowerowe "ROMET"
Krywałd "ERG" S.A.
Huta Łaziska
Huta Bankowa
Huta 1-go Maja
Huta Szkła Kunice
Zakłady Metalurgiczne Trzebinia
Walcownia Metali "Dziedzice"
Zakłady Górniczo - Hutnicze "BOLESŁAW"
Zakłady Gumowe Górnictwa
Fabryka Osi Napędowych
MOSTOSTAL
Elektromontaż 3
POLIFARB Cieszyn S.A.
POLAM Kostuchna
PKM Katowice
WEDEL S.A.
Instytut Informatyki Teoretycznej i Stosowanej
Centrum Naukowo Techniczne Kolejnictwa
Instytut Spawalnictwa
PRZEDSIĘBIORSTWO PRODUKCYJNO-USŁUGOWE INNOWACJI TECHNICZNYCH JUNISOFTEX



HEWLE
PACKAR

Niezwykłe.
Drukarki HP, które możesz zatrzymać w pełnym biegu.

Znane z ogromnej siły przetrwania, potrafią znieść trzęsienia ziemi, pożary i powodzie. Żywioły są wobec nich bezsilne. Nawet mrówki faraona nie są ich w stanie zatrzymać. Drukarki HP wykonują swoje zadania do końca, w każdych warunkach.
A teraz uwaga!
Nowa HP 5 potrafi więcej. Silna jak byk ale łagodna jak baranek. Zaopatrzona w specjalny przycisk "anulowanie pracy", gotowa zatrzymać się w każdej chwili podczas pracy za dotknięciem Twojego palca. To niezwykła zaleta dla użytkownika - może on własnoręcznie zatrzymać proces drukowania.
I ogromna ulga dla administratorów sieci - koniec z telefonami od zrozpaczonych użytkowników.
HP LaserJet 5 oprócz przycisku "anulowanie pracy" posiada ulepszoną tablicę kontrolną. Wyróżnia się też nowymi rozwiązaniami typu HP FontSmart, optional Flash, Windows 95 capability i Optional InfraRed adaptor.

Wszystko to sprawia, że HP LasertJet 5 jest godną - a co więcej, tańszą - następczynią HP LaserJet 4 Plus. Przy tym jest wyjątkowo prosta w obsłudze dla całego zespołu użytkowników. Zaawansowana technologia oprogramowania i oprzyrządowanie sprawia, że HP LaserJet 5 błyskawicznie reaguje na twoje polecenia. Posiada procesor 33 MHz i biegle włada naszym nowym językiem PCL 6. Dostępna jest w trzech różnych wariantach.
Network capable HP LaserJet 5.
LaserJet 5N dla sieci Ethernet 10 Base T.
LaserJet 5M dla zróżnicowanych środowisk.
Wszystkie one gwarantują doskonałą operatywność, redukują koszty bieżące i ..... i ...

HP Support Pack

dla wymagających
To serwis

O tak, tu można by się rozpisywać.
Ale czasami warto się na chwilę zatrzymać wśród biegu wydarzeń i zastanowić. Warto ją mieć.
Ten zakup procentuje.

DRUKARKI HEWLETT PACKARD